UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR DE LEGANÉS
INGENIERÍA DE TELECOMUNICACIÓN
DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

MÉTODOS DE AGRUPAMIENTO EN EL CÁLCULO DE DISTANCIAS ENTRE USUARIOS EN UN SISTEMA DE RECOMENDACIÓN BASADO EN ALGORITMOS DE FILTRADO COLABORATIVO

Proyecto Fin De Carrera

Autor: Felipe Ibáñez Vázquez
Tutor: Dr. Emilio Parrado Hernández
Leganés, 2011

A mi madre,

a Sara,

a mi hermana,

a mi padre,

a mi abuelo,

y a todas las personas que me han ayudado a llegar hasta aquí,

muchísimas gracias a todos.

Resumen

Los sistemas de recomendación tratan de automatizar los procesos sociales por los que habitualmente se recurre a las opiniones de terceros más expertos cuando se quiere tomar una decisión para adquirir algo sin tener la suficiente información para ello. Generalmente son decisiones sencillas en el entorno de la vida cotidiana, como qué libro leer, qué película ver, qué música escuchar o qué lugares visitar.

Unos de los tipos de sistemas de recomendación que existen son los basados en filtrado colaborativo. El proceso en estos sistemas consiste en recomendar ítems que han gustado a usuarios con gustos similares. Para ello, se tienen que recopilar en una base de datos las preferencias sobre los elementos de cada usuario para luego poder calcular las recomendaciones. A la hora de diseñar un sistema de recomendación mediante filtrado colaborativo hay que tener en cuenta la forma de almacenar las votaciones que se dan a los ítems por los usuarios del sistema, garantizar la privacidad y el tratamiento eficiente de la información e intentar dar soluciones a los problemas inherentes a estos sistemas.

En este proyecto fin de carrera se realiza un trabajo sobre las técnicas de filtrado colaborativo describiendo los algoritmos y métricas más empleados, así como las ventajas e inconvenientes de estos sistemas de recomendación.

El trabajo experimental es la implementación de un sistema de recomendación con técnicas de filtrado colaborativo donde modificamos el cálculo de distancias entre usuarios empleando métodos de agrupamiento. Para ello, tomaremos como base el toolkit implementado por el Doctor Guy Lebanon: *C/Matlab Toolkit for Collaborative Filtering* [1], y evaluaremos el impacto de utilizar dichos algoritmos de clustering en una plataforma ya creada sobre una base de datos real; la base de datos EachMovie del grupo de investigación GroupLens.

Palabras clave: *sistemas de recomendación, filtrado colaborativo, matriz de similitud, agrupamiento.*

Índice General

Resumen 5

Índice General 7

Índice de Figuras 9

1 INTRODUCCIÓN 13

1.1 Motivación 13

1.2 Objetivos 15

1.3 Estructura de la memoria 16

2 DESCRIPCIÓN DEL PUNTO DE PARTIDA 19

2.1 Sistemas de Recomendación y Filtrado Colaborativo 19

2.1.1 Tipos de Sistemas de Recomendación mediante Filtrado Colaborativo 21

2.1.2 Tipos de algoritmos 22

2.1.3 Medidas de distancia o similitud 23

2.1.4 Ventajas de los Sistemas de Recomendación con Filtrado Colaborativo 25

2.1.5 Desventajas de los Sistemas de Recomendación con Filtrado Colaborativo 25

2.1.6 Algunos Recursos disponibles en la red 27

2.2 Descripción de la plataforma utilizada 29

2.2.1 Ejemplo de Uso 30

3 DESCRIPCIÓN DE LA PROPUESTA 35

3.1 Detalle del trabajo realizado 35

3.2 Métodos Clustering 37

3.2.1 K-medias 37

3.2.2 Kernel K-medias 39

3.2.3 Mezcla de Gaussianas entrenadas con EM 41

4 VALIDACIÓN EXPERIMENTAL 43

4.1 Base de Datos 43

4.2 Medidas de Evaluación 44

4.2.1 Mean Absolute Deviation 44

4.2.2 Mean Square Error 45

4.2.3 Ranked Evaluation 45

4.3 Escenarios 46

4.3.1 K-medias ponderado 47

4.3.2 Mezcla de Gaussianas entrenadas con EM 47

4.3.3 Kernel K-medias 47

4.4 Presentación y discusión de resultados 48

4.4.1 Mean Absolute Deviation 48

4.4.2 Mean Square Error 57

4.4.3 Ranked Evaluation 64

5 CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO 73

5.1 Conclusiones 73

5.2 Líneas Futuras de Trabajo 74

6 PRESUPUESTO 75

6.1 Costes de personal 75

6.2 Costes de material 76

6.3 Presupuesto total 76

7 BIBLIOGRAFÍA 77

Índice de Figuras

Figura 2.1: Ejemplo de matriz de valoraciones 22
Figura 2.2 Diagrama de una ejecución de un sistema de recomendación con filtrado colaborativo mediante algoritmos basados en memoria usando el toolkit "C/Matlab Toolkit for Collaborative Filtering" 33
Figura 3.1 Pseudocódigo del algoritmo K-medias 38
Figura 3.2 Pseudocódigo del algoritmo Kernel K-medias 40
Figura 3.3 Pseudocódigo del algoritmo Mezcla de Gaussianas entrenadas con EM 42
Figura 4.1 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 5% del total de usuarios como centroides 48
Figura 4.2 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 5% del total de usuarios como centroides 49
Figura 4.3 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 10% del total de usuarios como centroides 50
Figura 4.4 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 10% del total de usuarios como centroides 51
Figura 4.5 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 20% del total de usuarios como centroides 52
Figura 4.6 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 20% del total de usuarios como centroides 52
Figura 4.7 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 30% del total de usuarios como centroides 53
Figura 4.8 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 30% del total de usuarios como centroides 54
Figura 4.9 Mean absolute deviation en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo basados en memoria usando el coeficiente de correlación de Pearson y Vector similarity 55

FIGURA 4.10 MEAN ABSOLUTE DEVIATION EN TODOS LOS ESCENARIOS SIMULADOS PARA LOS SISTEMAS DE RECOMENDACIÓN MEDIANTE FILTRADO COLABORATIVO UTILIZANDO UN 5% DEL TOTAL DE USUARIOS COMO CENTROIDES Y LOS MÉTODOS DE CLUSTERING....56
FIGURA 4.11 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 5% DEL TOTAL DE USUARIOS COMO CENTROIDES 57
FIGURA 4.12 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 5% DEL TOTAL DE USUARIOS COMO CENTROIDES....58
FIGURA 4.13 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 10% DEL TOTAL DE USUARIOS COMO CENTROIDES 59
FIGURA 4.14 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 10% DEL TOTAL DE USUARIOS COMO CENTROIDES....59
FIGURA 4.15 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 20% DEL TOTAL DE USUARIOS COMO CENTROIDES 60
FIGURA 4.16 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 20% DEL TOTAL DE USUARIOS COMO CENTROIDES....61
FIGURA 4.17 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 30% DEL TOTAL DE USUARIOS COMO CENTROIDES 62
FIGURA 4.18 MEDIDA MEAN SQUARE ERROR MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 30% DEL TOTAL DE USUARIOS COMO CENTROIDES....62
FIGURA 4.19 MEDIDA MEAN SQUARE ERROR EN TODOS LOS ESCENARIOS SIMULADOS PARA LOS SISTEMAS DE RECOMENDACIÓN MEDIANTE FILTRADO COLABORATIVO BASADOS EN MEMORIA USANDO EL COEFICIENTE DE CORRELACIÓN DE PEARSON Y VECTOR SIMILARITY....63
FIGURA 4.20 MEDIDA MEAN SQUARE ERROR EN TODOS LOS ESCENARIOS SIMULADOS PARA LOS SISTEMAS DE RECOMENDACIÓN MEDIANTE FILTRADO COLABORATIVO UTILIZANDO UN 30% DEL TOTAL DE USUARIOS COMO CENTROIDES Y LOS MÉTODOS DE CLUSTERING 64
FIGURA 4.21 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 5% DEL TOTAL DE USUARIOS COMO CENTROIDES 65
FIGURA 4.22 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 5% DEL TOTAL DE USUARIOS COMO CENTROIDES....66
FIGURA 4.23 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 10% DEL TOTAL DE USUARIOS COMO CENTROIDES 67

FIGURA 4.24 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 10% DEL TOTAL DE USUARIOS COMO CENTROIDES 67
FIGURA 4.25 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 20% DEL TOTAL DE USUARIOS COMO CENTROIDES 68
FIGURA 4.26 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 20% DEL TOTAL DE USUARIOS COMO CENTROIDES 69
FIGURA 4.27 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL NÚMERO DE USUARIOS ACTIVOS, VARIANDO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS POR LOS USUARIOS ACTIVOS Y USANDO UN 30% DEL TOTAL DE USUARIOS COMO CENTROIDES 70
FIGURA 4.28 MEDIDA RANKED EVALUATION MANTENIENDO FIJO EL PORCENTAJE DE PELÍCULAS VISTAS/PREDICHAS, VARIANDO LA PROPORCIÓN DE USUARIOS ACTIVOS Y USANDO UN 30% DEL TOTAL DE USUARIOS COMO CENTROIDES 70
FIGURA 4.29 MEDIDA RANKED EVALUATION EN TODOS LOS ESCENARIOS SIMULADOS PARA LOS SISTEMAS DE RECOMENDACIÓN MEDIANTE FILTRADO COLABORATIVO BASADOS EN MEMORIA USANDO EL COEFICIENTE DE CORRELACIÓN DE PEARSON Y VECTOR SIMILARITY 71
FIGURA 4.30 MEDIDA RANKED EVALUATION EN TODOS LOS ESCENARIOS SIMULADOS PARA LOS SISTEMAS DE RECOMENDACIÓN MEDIANTE FILTRADO COLABORATIVO UTILIZANDO UN 30% DEL TOTAL DE USUARIOS COMO CENTROIDES Y LOS MÉTODOS DE CLUSTERING 72
FIGURA 6.1 PRESUPUESTO PARA LOS COSTES DE PERSONAL 75
FIGURA 6.2 DESGLOSE DEL PRESUPUESTO DE LOS COSTES DE PERSONAL 75
FIGURA 6.3 PRESUPUESTO PARA LOS COSTES DE MATERIAL 76
FIGURA 6.4 PRESUPUESTO PARA LOS COSTES TOTALES 76

1 INTRODUCCIÓN

En este primer capítulo vamos a proporcionar una visión global del contenido de este Proyecto de Fin de Carrera. Inicialmente se darán a conocer las motivaciones que nos han hecho interesarnos por este campo de investigación. A continuación, veremos cuáles han sido los principales objetivos que nos hemos marcado con la realización de este trabajo. Al final del capítulo se hará una breve descripción del contenido de la memoria del proyecto.

1.1 Motivación

En la época donde Internet todavía no estaba tan desarrollado como hoy en día, la información que disponía un usuario sobre un producto/servicio y los medios para acceder a él estaba limitado. En los últimos años la situación ha pasado a ser justo la contraria, se ha experimentado un enorme crecimiento tanto en contenidos disponibles, información acerca de ellos y sus posibles alternativas, que el usuario final puede sentirse desbordado y no saber que creaciones le interesan y cuáles no. Este problema se ha visto agravado desde que principalmente realizamos búsquedas a través de Internet, redes P2P, tiendas online etc.

La información sobre contenidos siempre ha existido, proporcionada por ejemplo por grandes superficies, videoclubs o venta por catálogo, pero no ha sido aprovechada para hacer sistemas de recomendación hasta que ha existido capacidad de cómputo, capacidad de almacenamiento y algoritmos de machine learning que la aprovecharan para ello.

Esta difícil gestión ha hecho que los sistemas de recomendación se hayan ido consolidando como potentes herramientas para ayudar a reducir la sobrecarga de información a la que nos enfrentamos en los procesos de búsqueda, porque ayudan a evaluar y filtrar el contenido. Estos sistemas generan recomendaciones a partir de las preferencias y opiniones dadas por otros usuarios, o bien a partir de las preferencias del usuario objeto de la recomendación, dando lugar a los Sistemas de Recomendación colaborativos y los no colaborativos o basados en contenidos.

Durante la última década, los motores de recomendación los podemos encontrar en gran parte de nuestro día a día. Casi sin darnos cuenta aparecen en el momento apropiado para

hacernos sugerencias mientras hacemos acciones tan cotidianas en Internet como compras on-line, navegar por nuestras redes sociales favoritas o mirar las últimas novedades de la cartelera.

Un buen motor de recomendación vale mucho dinero. Según un informe de Forrester, un tercio de los clientes que recibe recomendaciones durante sus compras on-line terminan comprando alguno de los productos recomendados [2].

Nuestro estudio se va a centrar en los Sistemas de Recomendación mediante Filtrado Colaborativo describiendo los aspectos más significativos. Nos centraremos en ellos porque a mi entender, este tipo de sistemas juegan un papel importante en las actuales redes sociales y la Web 2.0. Algunos ejemplos son:

- Tiendas on-line. Partiendo de un producto se recomiendan otros productos que han interesado a los usuarios que compraron dicho producto.

  Una tienda on-line que implementa este tipo de recomendaciones es Amazon. Utiliza un sistema de recomendaciones que recurre tanto a los artículos que el usuario ha adquirido con anterioridad, como a los que, de alguna forma le han interesado al visitarlos o añadirlos a su lista de deseos. Amazon utiliza tres criterios principales: el usuario, el entorno social y los productos. De esta manera, basa sus recomendaciones en el comportamiento individual y en el de otros usuarios ante el mismo tipo de producto.

  Amazon puso de moda la frase: "Customers Who Bought This Item Also Bought…"

- Recomendaciones acerca de contenido cultural. Cada vez que un usuario escucha una canción/ve una película/lee un libro se envía su información a la base de datos del sistema, el cual las utiliza para generar nuestras recomendaciones.

  Nexflix es la compañía de alquiler con el mayor catálogo de películas en Estados Unidos. Su modelo de negocio es cobrar suscripciones a un servicio de alquiler en él que los clientes eligen los títulos que quieren por Internet y después reciben por correo.

  El sistema de recomendaciones es clave para Nexflix, ya que sus suscriptores tienen un amplio catálogo donde elegir, y sin un sistema que ayude al usuario sería un caos entre desaciertos y búsquedas de información. Por este motivo, el sistema de recomendaciones es lo que mantiene vivo el valor que perciben los clientes de Netflix, precisamente porque actúa de filtro ante un gran catálogo de películas y de audiencia. La gente simplemente valora las que ha visto y el sistema le recomienda otras afines a sus gustos.

  Pandora es un sistema de recomendación de música. Contiene información acerca de características y atributos de cada uno de sus temas musicales. Las personas ofrecen información al sistema sobre sus canciones y artistas favoritos y luego éste les recomienda aquellas canciones cuyas características almacenadas concuerden con sus preferencias.

- Búsqueda en comunidades. En las redes sociales se evalúan los gustos del usuario para generar una lista de posibles vecinos con similares intereses.

  Redes como Facebook son capaces de mostrar en la página personalizada de cada usuario los grupos, hilos de conversación o personas de su interés basado en el comportamiento del resto de sus contactos.

- Filtrado de noticias. Construcción de un recomendador de noticias en función del perfil de noticias que normalmente cada usuario consulta.

1.2 Objetivos

El filtrado colaborativo consiste en recomendar elementos que han gustado a usuarios con preferencias similares, basándose únicamente en la puntación que éstos asignan a los ítems del sistema. Para ello las personas puntúan los elementos de acuerdo a sus intereses (a medida que esto sucede se va generando su perfil) y a partir de ellos se calculan las recomendaciones.

Algunos enfoques se basan en calcular la similitud entre usuarios, y a partir de sus preferencias se obtienen los elementos a recomendar. Para cada usuario se crea un conjunto de "vecinos cercanos": personas cuyas evaluaciones tienen grandes semejanzas a las del usuario que solicita la recomendación. Los resultados para los elementos no calificados por él, se predicen en base a la combinación de puntuaciones conocidas de los vecinos cercanos.

Entonces el procedimiento general aplicado en estos sistemas se resume de la siguiente forma:

a. Los usuarios expresan sus valoraciones sobre elementos del sistema, generalmente mediante una escala numérica.
b. A partir de esa información, se intenta predecir la puntuación que daría el usuario que solicita la recomendación (usuario activo), a los elementos del sistema no conocidos hasta el momento por él.
c. De las predicciones calculadas se seleccionan los elementos con valores más altos para realizar la recomendación.

Existen muchas maneras de generar las predicciones mencionadas, desde diferentes enfoques y basando los cálculos en diferentes algoritmos.

El objetivo principal de este proyecto es la introducción de métodos de agrupamiento en sistemas de recomendación basados en algoritmos de filtrado colaborativo. Nos centraremos

en probar diferentes métodos de agrupamiento a la hora del cálculo de las distancias entre usuarios.

Para ello, es necesario tener una vista previa de los sistemas de recomendación, los algoritmos de filtrado colaborativo y de las técnicas de agrupamiento. Se hará un repaso del estado del arte de los sistemas de recomendación, y se explicarán en detalle los algoritmos de agrupamiento que usaremos. Estos son: Kernel K-medias [3], Soft K-medias y Modelos de Mezcla de Gaussianas entrenadas con EM [4].

Experimentalmente vamos a comparar la posibilidad de descubrir grupos con distinta forma, la capacidad de manejar datos de gran dimensionalidad, la facilidad de interpretación o la posibilidad de tener un conocimiento mínimo del entorno para determinar los parámetros de entrada de los algoritmos de agrupamiento.

1.3 Estructura de la memoria

La organización del contenido de la memoria del presente Proyecto Fin de Carrera se ha llevado a cabo de la siguiente manera:

- Capítulo 1: Introducción.

  En este primer capítulo se proporciona una visión global del contenido de este proyecto de Fin de Carrera, se dan a conocer las motivaciones que nos han hecho interesarnos por este campo, y se ven cuáles han sido los principales objetivos que nos hemos marcado.

- Capítulo 2: Descripción del Punto de Partida.

  En este capítulo vamos a exponer el punto de partida del trabajo realizado. Se hará una presentación del estado del arte sobre filtrado colaborativo, tipos de sistemas de recomendación, ventajas, inconvenientes y medidas de similitud empleadas.

  En la segunda parte, vamos a facilitar un listado de los recursos disponibles en la red que tenemos para trabajar con sistemas de recomendación, y haremos una descripción en detalle del entorno de trabajo elegido para hacer nuestros experimentos. Como plataforma base hemos usado el toolkit implementado por el Doctor Guy Lebanon: *C/Matlab Toolkit for Collaborative Filtering* [1], el cual nos proporciona un conjunto de funciones de Matlab y C que implementan varios métodos de filtrado colaborativo.

- Capítulo 3: Propuesta.

  En el apartado tercero detallaremos los algoritmos de agrupamiento propuestos para el cálculo de distancias entre usuarios dentro de sistemas de recomendación mediante filtrado colaborativo y profundizaremos en explicar con detalle el trabajo de investigación y desarrollo realizado.

- Capítulo 4: Validación Experimental.

  En el capítulo cuarto de este proyecto pasamos a mostrar el estudio experimental comparativo realizado, donde presentaremos los experimentos simulados y sus resultados.

  Describiremos la base de datos empleada, las medidas de evaluación con las que van a hacerse las comparaciones entres sistemas de recomendación y los escenarios con los que se han ejecutado los experimentos propuestos. Se hará una presentación y discusión de los resultados.

- Capítulo 5: Conclusiones y Líneas Futuras de Trabajo.

  En este último capítulo se presentan las conclusiones finales consecuencia del trabajo realizado y la explicación de posibles líneas que quedan abiertas.

2 DESCRIPCIÓN DEL PUNTO DE PARTIDA

En este capítulo vamos a exponer el punto de partida del trabajo realizado en este proyecto. En primer lugar, se va a hacer una presentación general de lo que es un sistema de filtrado colaborativo, y a continuación veremos con más detalle el estado del arte sobre esta técnica presente en la literatura. Se hablará de los tipos de sistemas de recomendación mediante filtrado colaborativo existentes, de los algoritmos que utilizan y de sus ventajas e inconvenientes. A continuación hablaremos de las dos medidas de similitud empleadas en los sistemas de recomendación que hemos realizado.

Para finalizar, vamos a facilitar un listado de los recursos disponibles en la red que tenemos para implementar sistemas de recomendación, y haremos una descripción en más detalle de la plataforma elegida para hacer nuestro trabajo experimental.

2.1 Sistemas de Recomendación y Filtrado Colaborativo

Un sistema de recomendación es un tipo específico de filtro de información, técnica que trata de presentar al usuarios ítems de información (películas, música, libros, noticias, páginas web) sobre las que el usuario está interesado. No todos los sistemas de recomendación son iguales, su diseño depende del dominio y de las características particulares de la base de datos. Para crear dicha base de datos se registran las interacciones entre usuarios y artículos. Adicionalmente, se puede tener acceso a los perfiles de los clientes o los productos, como edad o descripción, respectivamente.

Los sistemas de recomendación se diferencian en la forma en la que analizan la información para construir la afinidad entre consumidores y artículos. Los basados en filtrado colaborativo analizan solamente el historial de interacciones, mientras que los basados en contenido utilizan los atributos del perfil.

El enfoque más intuitivo que surge a la hora de crear un sistema de recomendación es que éste analice la relación entre el contenido de los elementos de interés del usuario y el contenido de los elementos del sistema, para así poder realizarle sugerencias. Como ejemplo de ello se puede pensar en una aplicación que recomienda libros donde la relevancia de los ítems a sugerir se determina comparando las descripciones de cada libro (autor, género, etc.) y las descripciones de los libros de interés del usuario. A este tipo de aplicaciones se les denomina "sistemas de recomendación basados en contenido", dado que el filtrado de la información se realiza analizando el contenido de los elementos.

Los sistemas que recomiendan elementos del gusto de los usuarios, cuando aplican la técnica de filtrado basado en contenido, cuentan con algunas limitaciones. Por un lado, los elementos que sugieren deben tener atributos que permitan describirlos. Con la tecnología actual y medios como música, fotografía, arte, video o elementos físicos, se dificulta esta tarea, ya que generalmente estos ítems no pueden ser analizados automáticamente debido a la dificultad de obtener un conjunto de atributos que permitan describirlos realmente. Las predilecciones de los usuarios pueden deberse a las sensaciones que provocan tales elementos y no a sus atributos. Por ejemplo, el agrado por una canción puede ser ocasionado por la satisfacción de escucharla, no influyendo en ello su género, autor, intérprete o frecuencia musical. Por otro lado, dadas las características de la técnica de filtrado basado en contenido, estos sistemas no permiten que el usuario descubra ítems de contenido totalmente distinto a los ya conocidos por él aunque puedan ser de su interés. Por ejemplo, en un sistema en que se recomienda películas teniendo en cuenta el género preferido por el usuario, si éste manifestó interés por películas de suspense, el sistema omitirá realizarle recomendaciones de películas de otros estilos distintos a suspense, sin sugerirle que experimente otros géneros.

Con el fin de paliar estas limitaciones surgen los sistemas de recomendación basados en filtrado colaborativo. En estos sistemas, el filtrado de información no se realiza analizando las características de los elementos a recomendar, sino que para ello se tiene en cuenta únicamente las valoraciones de todos los usuarios sobre los ítems del sistema. El uso típico de esta técnica es en sistemas de recomendación de libros, música y películas.

Un sistema de recomendación mediante algoritmos de filtrado colaborativo usa la información conocida sobre las preferencias de otros usuarios para realizar la recomendación al usuario que la precise. Los sistema de recomendación colaborativos identifican usuarios cuyas preferencias sean similares a las de otros usuarios dados y recomiendan a los primeros los elementos que hayan satisfecho a los segundos. De esta forma, si dos usuarios *U1* y *U2* tienen las mismas preferencias y al usuario *U1* le ha satisfecho un ítem *i*, probablemente este ítem también satisfaga al usuario *U2* por lo que deberíamos recomendárselo. Por ello, en estos sistemas de recomendación la definición de medidas de similitud entre preferencias es un punto crítico. La situación puede ser representada como una matriz de usuarios por ítems, donde posición i,j celda representa la valoración de un usuario i con respecto a un ítem j concreto. Así visto, el problema consiste en predecir valores para las posiciones que estén vacías.

Los sistemas de recomendación colaborativos tienen importantes ventajas con respecto a los no colaborativos:

- Dan un mayor soporte para el filtrado de ítems cuyo contenido no es fácil de analizar por procesos automatizados.
- La posibilidad de filtrar ítems basándose en su calidad o preferencias.
- La posibilidad de realizar recomendaciones válidas, pero que no esperábamos, lo cual puede resultar de gran utilidad.

Sin embargo, también presentan inconvenientes, como por ejemplo que no trabajan bien a la hora de filtrar información para necesidades de contenido específicas, o cuando el número de usuarios es bajo. Por ello, en muchas ocasiones la mejor opción es adoptar un enfoque híbrido entre colaborativo y no colaborativo y de esta forma disfrutar de las ventajas de ambos.

Aun así, nosotros nos centraremos únicamente en los sistemas de recomendación basados en filtrado colaborativo porque en este proyecto buscamos estudiar el efecto que produce la inserción de métodos de agrupamiento en el cálculo de distancias entre usuarios en un sistema de recomendación basado en algoritmos de filtrado colaborativo.

2.1.1 Tipos de Sistemas de Recomendación mediante Filtrado Colaborativo

Los dos tipos de sistemas de recomendación mediante filtrado colaborativo que vamos a presentar son los basados en memoria y los basados en modelo.

2.1.1.1 Basados en memoria, o algoritmos de vecinos cercanos

Utilizan toda la base de datos de elementos y usuarios para generar predicciones. Primeramente emplean técnicas estadísticas para encontrar a vecinos, es decir, usuarios con un historial de valoraciones sobre los elementos similar al usuario actual. Una vez que se ha construido una lista de vecinos se combinan sus preferencias para generar una lista con los N elementos más recomendables para el usuario actual.

Entre sus inconvenientes se encuentra la necesidad de disponer de un número mínimo de usuarios con un número mínimo de predicciones cada uno, incluido el usuario para el que se pretende realizar la recomendación.

2.1.1.2 Basados en Modelo

Desarrollan primero un modelo de los ratings del usuario. Tratan el problema como un problema de predicción estadística y calculan el valor esperado para cada ítem en función de los ratings anteriores. Para ello se utilizan distintos algoritmos de aprendizaje, clustering o redes neuronales como las Redes de Funciones de Base Radial (RBFN). Por ejemplo, utilizando

clustering se trata de clasificar a un usuario en particular dentro de una clase de usuarios y a partir de ahí se estiman las probabilidades condicionadas de esa clase hacia los elementos a evaluar.

En general, ante las consultas responden más rápido que los basados en memoria, pero por contra necesitan de un proceso de aprendizaje intensivo.

2.1.2 Tipos de algoritmos

Una vez descritos los tipos de sistemas de recomendación mediante filtrado colaborativo veremos los algoritmos posibles que pueden implementar.

2.1.2.1 Algoritmos basados en vecinos cercanos.

Fueron los primeros algoritmos de filtrado colaborativo en implementarse. En primer lugar es necesario medir los parecidos de todos los usuarios con el usuario actual. Para ello pueden utilizarse distintas medidas. Funcionan seleccionando un conjunto apropiado de usuarios, según la similitud de los mismos con respecto al usuario activo, y usan las valoraciones de dichos usuarios para generar la valoración del usuario activo. Concretamente, los tres pasos a seguir para realizar esto son los siguientes:

1. Medir la similitud de todos los usuarios con respecto al usuario activo.
2. Seleccionar un subconjunto de usuarios cuyas valoraciones se van a usar y por tanto, tendrán influencia en la generación de la predicción para el usuario activo.
3. Normalizar las puntuaciones de los distintos usuarios y calcular una predicción a partir de algún tipo de combinación pesada de las puntuaciones asignadas al ítem por los usuarios seleccionados en el paso anterior.

En la tabla 1 podemos ver un ejemplo de matriz que representa valoraciones de usuarios con respecto a una serie de películas.

	Terminator	Pretty woman	El caballero oscuro	El último mohicano
Felipe	**5**	**2**	**5**	**4**
Sara	**2**	**5**		**3**
Alberto	**3**	**3**	**2**	**2**
Paula	**¿?**	**1**	**4**	**5**

Figura 2.1: Ejemplo de matriz de valoraciones

Podemos predecir que a Paula le gustará la película Terminator. Observamos que Felipe es el que tiene preferencias más parecidas a Paula, puesto que ambos tienen unas valoraciones muy similares de las películas que ya han visto. Por tanto la valoración de Felipe sobre la película Terminator tendrá gran influencia en la predicción que hagamos a Paula sobre dicha película. Por el contrario, Sara y Alberto tienen opiniones más dispares con respecto a Paula, por lo que tendrán una influencia mucho menor en las recomendaciones que se hagan a dicho usuario.

2.1.2.2 Algoritmos basados en elementos.

En lugar de buscar similitudes entre usuarios buscan cercanías entre elementos. El procedimiento consiste en seleccionar los elementos que un usuario determinado ha votado y después comprobar cómo de similar es cada uno del resto de los elementos del sistema, para terminar recomendando los más parecidos. Existen distintas formas de evaluar la similitud entre elementos pero el procedimiento genérico consiste en tomar dos elementos x1, x2 y después calcular su similitud a partir de todos los usuarios que han votado ambos elementos. En teoría es la misma aproximación que la que se tenía con algoritmos basados en vecinos cercanos.

La ventaja es que en el caso de los elementos la similitud entre ellos es menos variable que la similitud entre usuarios, lo que permite pre-computar estas similitudes y hace el proceso mucho más rápido.

2.1.3 Medidas de distancia o similitud

Las medidas de similitud o de distancia son expresiones matemáticas que permiten resumir en un número el grado de relación entre dos entidades, midiendo sobre la base de semejanza o la desigualdad entre la cualidad o la cantidad de sus atributos, o ambas. Estas medidas son fundamentales en la definición de métodos de agrupamiento y han sido estudiadas desde la década de los 50, en campos tan variados como la psicología o el tratamiento de imágenes, así como en temas de filtrado colaborativo para sistemas de recomendación. Las medidas de similitud más usuales en este último campo son el coeficiente de correlación de Pearson [5] y el vector de similitud o vector similarity [5].

2.1.3.1 Coeficiente de correlación de Pearson

Esta medida de similutud es una de las más utilizadas en el ámbito del filtrado colaborativo. El coeficiente de correlación de Pearson entre un usuario a y otro usuario i, se puede expresar como:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

donde j indica cuales son aquellos elementos para los que ambos usuarios han registrado votos.

Una vez se tienen los pesos de correlación de cada algoritmo hay que saber como de confiables son estos pesos. Es posible tener un alto grado de correlación con vecinos con los que se comparten pocos elementos valorados por el usuario actual pero con igual valoración. El uso de estos pesos proporciona unas estimaciones malas, puesto que para tener una idea real de la correlación son necesarios cuantos más votos compartidos mejor. En los casos de pocas muestras es recomendable disminuir el factor de correlación en función del número de votos compartidos.

Para intentar mejorar más los pesos de la correlación entre usuarios se puede entrar a trabajar con la varianza de los elementos que cada usuario ha votado. Si un elemento es votado positivamente por un gran porcentaje del conjunto de usuarios, el que dos usuarios compartan esa votación dice poca información acerca de la correlación entre usuarios. Lo contrario pasa en el caso de elementos que sean votados positiva o negativamente por pocos usuarios. Para tener en cuenta este hecho se añade a la formula de la correlación de Pearson un termino con la varianza del elemento.

2.1.3.2 Vector de Similitud o Vector Similarity

En el campo de recuperación de información, la similitud entre dos documentos se mide a menudo considerando cada documento como un vector de frecuencias de palabras y calculando el coseno del ángulo formado por los dos vectores de frecuencia [6]. Se puede adoptar este formalismo para sistemas de filtrado colaborativo, donde los usuarios toman el papel de los documentos, los títulos asumen el papel de las palabras y los votos toman el papel de las frecuencias de la palabra. Hay que apreciar que en estos algoritmos, los votos indican una preferencia positiva, no teniéndose en cuenta los votos negativos, y los artículos no observados reciben un voto nulo. Los pesos correspondientes son entonces:

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

donde los términos al cuadrado que aparecen en el denominador sirven para realizar una normalización de los votos. De esta manera los usuarios que votan más títulos no serán a priori más similares a otros usuarios. Se pueden utilizar otros esquemas de normalización, como la suma absoluta y el número de votos.

2.1.4 Ventajas de los Sistemas de Recomendación con Filtrado Colaborativo

Las principales ventajas de los sistemas de recomendación que utilizan técnicas de filtrado colaborativo son:

- Se pueden utilizar en numerosas aplicaciones ya que el FC puede aplicarse a cualquier tipo de ítem o producto: documentos, música, películas, libros, etc.

- Dan un mayor soporte para el filtrado de ítems cuyo contenido no es fácil de analizar por procesos automatizados.

- Otorgan la posibilidad de filtrar ítems basándose en su calidad o preferencias de los usuarios del sistema.

- Permiten realizar recomendaciones válidas que no eran esperadas, lo que puede resultar de gran utilidad.

- La gran ventaja de los sistemas de recomendación colaborativos sobre los basados en contenido está relacionado con la habilidad de "cruzar géneros" (cross-genre recommendations) o "salir de la caja" (outside the box). Estos sistemas tienen la capacidad para recomendar artículos distintos de los que el usuario ha evaluado, pero que pueden ser de interés.

2.1.5 Desventajas de los Sistemas de Recomendación con Filtrado Colaborativo

Existen varios inconvenientes en la aplicación de filtros colaborativos en sistemas de recomendación. Mostramos los principales a continuación:

- Problema de Cold-Start

Se produce cuando llega un usuario o un ítem nuevo al sistema.

Cuando un usuario llega al sistema, no es posible hacerle recomendaciones hasta que su perfil esté lo suficientemente determinado como para encontrarle su grupo de vecinos cercanos. Además, si los gustos del usuario son poco comunes, podría no ser fácil encontrarle un conjunto de vecinos. Esto hace notar que las recomendaciones dependen directamente del número y variedad de usuarios en el sistema. En la mayoría de los casos se ataca este problema exigiéndoles a los nuevos usuarios que tomen postura sobre una serie de elementos, de manera de poder determinar un perfil inicial como para poder realizarle una primera recomendación.

También es considerado un problema de Cold-Start cuando la llegada no es por parte de un usuario, sino de un nuevo ítem. Sin embargo no se ha encontrado en la bibliografía consultada sugerencias sobre cómo resolver el problema del Cold-Start en esta situación. En este caso no se tendría ninguna valoración por parte de los usuarios sobre ese ítem, lo cual imposibilita su recomendación.

- Problema de Dispersión o Sparsity

Este problema surge cuando el número de usuarios es pequeño en relación al volumen de información en el sistema. Es decir, se debe a que cada usuario sólo califica a un número relativamente reducido de ítems. Por tanto la matriz de calificaciones usuario-ítems es una matriz dispersa, con numerosos valores nulos. Como resultado de lo anterior, la probabilidad de encontrar perfiles similares es habitualmente baja y por tanto decrece la cantidad de recomendaciones posibles significativamente.

- Problema de Escalabilidad

Si el número de elementos en el sistema aumenta, se ve incrementado el tiempo de cómputo al calcular el conjunto de vecinos cercanos. Por lo tanto, se observa que este tipo de sistemas tienen problemas de escala ya que al aumentar el número de ítems en el sistema, se hace difícil mantener un rendimiento razonable en las recomendaciones. Posibles soluciones a este problema pueden ser el cálculo periódico de las recomendaciones o el particionamiento del conjunto de elementos.

- Confianza entre usuarios

En las recomendaciones del tipo colaborativo, se pude dar el caso de que existan usuarios mal intencionados que intentan corromper el sistema de recomendación asignando votos muy altos a ítems que son de su interés con el objetivo que sean recomendados. Massa y Avesani en [7] proponen la creación de una “red de confianza” para solventar el problema. La idea es que un usuario, además de puntuar los ítems del sistema, también exprese su confianza en las valoraciones que han hecho otros usuarios. Así se consigue basar las recomendaciones únicamente con las valoraciones de usuarios en los que se confía.

- Problema de la “Oveja Gris”

Los sistemas de recomendación colaborativos trabajan mejor para un usuario que encaja en un grupo de usuarios numeroso de gustos similares. Pero en ocasiones existen usuarios cuyos perfiles están entre las fronteras de dos grupos de usuarios, no encajando en ninguno de ellos totalmente. A este tipo de usuarios que utilizan el sistema de recomendación se les denomina “ovejas grises”. En este caso de la aparición de dichos usuarios se hace difícil determinan para los mismos una recomendación adecuada.

- Problema de la Sinonimia

Este problema se produce por la escasez de cualquier forma de interpretación semántica. Es decir, los sistemas de recomendación colaborativa sólo confían en las valoraciones de los ítems para recomendarlos sin tener en cuenta los datos descriptivos. De esta manera ítems similares no se tratarán de igual forma por el sistema a la hora de realizar las recomendaciones a los usuarios.

- Problema de Subjetividad

Por último aparece el problema de la subjetividad con respecto a la naturaleza de los ratings o calificaciones otorgadas a los ítems por el usuario.

2.1.6 Algunos Recursos disponibles en la red

En este apartado vamos a recopilar parte de los materiales disponibles en Internet para empezar a trabajar en sistemas de recomendación mediante filtrado colaborativo.

2.1.6.1 Software

- C/Matlab Toolkit for Collaborative Filtering:

  Toolkit de Matlab que incluye algunos algoritmos de filtrado colaborativo [8]

  http://www.cc.gatech.edu/~lebanon/software/CFtoolbox/
- Apache Mahout:

  Motor de filtros colaborativos escrito en JAVA.

  http://mahout.apache.org/
- CoFE:

  Otro motor de filtrado colaborativo basado en Java.

  http://eecs.oregonstate.edu/iis/CoFE/
- Suggest Top-N recommendation engine:

  Implementa algoritmos de filtrado colaborativo item-based y user-based.

  http://www-users.cs.umn.edu/~karypis/suggest/

2.1.6.2 Data sets

Existen dos formas de recoger las preferencias de los usuarios. Por *extensión* se refiere a información que se tenga sobre las experiencias pasadas del usuario con respecto a los ítems encontrados. Es lo que también conocemos como *navegación implícita* pues el usuario no es consciente de estos seguimientos. Por *información expresada intencionalmente* se entiende alguna especificación de los ítems deseados por los usuarios. También se le llama *navegación explícita* y consiste en que el usuario expresa intencionalmente al sistema de recomendación información sobre sus preferencias.

La recolección de preferencias de forma explícita es simple de realizar pues consiste únicamente en ofrecer la posibilidad de puntuar los ítems del sistema. Sin embargo, para lograr esto se requiere un esfuerzo importante por parte de los usuarios, dado que les puede llevar mucho tiempo realizar estas acciones.

Para recoger las preferencias de forma implícita el sistema debe ser capaz de estudiar el comportamiento del usuario en su utilización, lo cual es claramente una tarea difícil. Además, es preciso considerar que las preferencias que se pueden obtener de esta forma son menos exactas dado que son estimadas por el sistema y no proporcionadas directamente por el usuario. Sin embargo, esta forma de recolección resulta más cómoda para las personas dado que no deben puntuar cada uno de los elementos.

Si bien la información explícita tiene una mayor exactitud y es más fácil de analizar por el sistema, la recolección implícita no deja de ser importante ya que facilita la tarea de las personas que utilizan la aplicación. Por lo tanto, muchos sistemas intentan recoger las preferencias de ambas formas, logrando complementar los esfuerzos de las dos partes (usuario y sistema).

2.1.6.2.1 Navegación explícita

- MovieLens: http://www.grouplens.org/node/73

  El data set de películas de GroupLens es el data set de referencia, utilizado en la mayoría de las publicaciones sobre filtros colaborativos. Actualmente el grupo de investigación de la universidad de Minnesota, GropuLens, tiene 2 bases de datos disponibles. La primera consiste en 100.000 puntuaciones para 1682 películas por 943 usuarios. La segunda consiste de aproximadamente 1 millón de votos anónimos para 3900 películas por 6040 usuarios. La puntuación que se le da a las películas es de 1-5.
- Wikilens: http://www.grouplens.org/node/425

  Data set genérico que permite a la comunidad de usuarios definir ítems y evaluarlos.
- Book-Crossing: http://www.grouplens.org/node/74

  Data set de libros extraído mediante rastreo en la comunidad Book-Crossing.

La base de datos BookCrossing (BX) fue recolectada por Cai-Nicolas Ziegler en cuatro semanas (Agosto – Setiembre de 2004) de la comunidad Book-rossingrnbaker. Esta contiene 278.858 usuarios anónimos pero con información geográfica, 1.149.780 votos con una puntuación de 0 a 10 de 271.379 libros.

- Jester joke: http://www.grouplens.org/node/75

  Data set de bromas. Ken Goldberg de la Universidad de California, Berkeley, también ha lanzado la base de datos del Sistema de Recomendación Jester Joke. Esta base de datos contiene 4.1 millones de votos sobre 100 chistes de 73.496 usuarios, recolectados entre abril de 1999 y mayo de 20034. Lo inconveniente de esta base de datos es la poca cantidad de ítems que contiene.
- Online Dating Data Set: http://www.ksi.ms.mff.cuni.cz/%7Epetricek/data/

  Data set procedente de una agencia de citas online.

2.1.6.2.2 Navegación implícita

- Audioscrobblers Music Play-list Data-sets: http://www.audioscrobbler.com/data/

  Data set procendente de la comunidad Web de Audioscrobblers. Se construye automáticamente a través de las canciones que los usuarios van escuchando en su reproductor de música.
- AOL Web search query: http://www.gregsadetsky.com/aol-data/

2.2 Descripción de la plataforma utilizada

Como plataforma base para llevar a cabo nuestros experimentos vamos a usar el toolkit implementado por el Doctor Guy Lebanon, actualmente profesor en el Grupo de Computación del Instituto Tecnologico de Georgia [1].

Este toolkit es un conjunto de funciones de Matlab y C que implementan varios métodos de filtrado colaborativo. Además de implementar varios algoritmos propuestos en la literatura reciente, también suministra funciones para la carga, la manipulación y la evaluación de métodos de filtrado colaborativo. Los módulos con la principal carga computacional se implementan en C como archivos de mex y se llaman desde Matlab, logrando así una computación más eficiente.

A continuación procedemos a llevar a cabo una breve explicación de los principales métodos que el toolkit nos ofrece:

- eachMovieReader.m - carga usuarios de la base de datos de EachMovie desde un documento de texto local en una matriz sparse de Matlab.

- splitUsers.m - divide al azar la matriz generada por eachMovieReader.m en dos matrices sparse para separar a los usuarios activos de los demás.
  En nuestros experimentos definimos como usuarios activos a aquellos que utilizamos como conjunto de test.

- EvalMemBasedtEachMovie.m – evalúa la predicción de los algoritmos basados en memoria.

- memoryBasedModels.c - calcula la matriz de similitud entre los usuarios activos y los demás usuarios. La función es compatible con las siguientes medidas de similitud: correlación de Pearson, vector de similitud, votación predeterminada y case amplification [5].

- predictPreferenceMemBased.m - calcula la preferencia predicha de los ítems para un usuario activo mediante la salida de la matriz de similitud memoryBasedModels.c.

- EvalAvgEachMovie.m - evalúa el método de predicción que predice la preferencia para todos los ítems como la preferencia promedio del usuario (basándose en los datos de entrenamiento).

- EvalConstEachMovie.m – evalúa el método de predicción que predice la preferencia media para todos los elementos para todos los usuarios.

- rankedEvalCF.m – método de evaluación por ranking que generaliza el método de evaluación propuesto en [9] a varios valores de preferencia.

- eachMovieComparisonScript.m – script principal que administra los experimentos que comparan el rendimiento de los diferentes métodos de CF en el conjunto de datos de EachMovie.

2.2.1 Ejemplo de Uso

En este apartado detallamos la funcionalidad del conjunto de herramientas a través de un ejemplo utilizando el toolkit base que nos proporciona Guy Lebanon. Este muestra la utilidad del toolkit en las diferentes etapas de predicción y evaluación de filtrado colaborativo.

El primer paso es leer el conjunto de datos y almacenarlo en memoria. Gracias al comando *eachMovieReader* podemos almacenar usuarios de la base de datos empleada, en nuestro caso EachMovie:

>>userVoteMat=eachMovieReader(300);

Trescientos es el número de usuarios que se leen de los ochenta mil de los que consta nuestro conjunto de datos. Leer toda la base de datos lleva varios minutos dependiendo del ordenador utilizado, y consume una gran cantidad de memoria física, por lo que se recomienda hacer las pruebas de concepto con un menor.

La salida *userVoteMat* es una matriz de tipo sparse en la que *userVoteMat(i,j)* es la votación almacenada por el usuario i de la película j. Si no hay voto se guarda un cero.

El siguiente paso es dividir el conjunto de datos en usuarios activos y usuarios no activos. El conjunto de estos últimos se emplea para proporcionar predicciones para la preferencia de los usuarios activos. Esta división es análoga a hacer un conjunto de entrenamiento y test en clasificación. La división elige algunas de las filas al azar y forman una matriz *activeMat* nueva, con las filas restantes se crea la matriz *otherMat*. Esto se puede hacer usando el comando *splitUsers*:

>>numNonActive=size(userVoteMat,1)-150;
>>[activeMat,otherMat]=splitUsers(userVoteMat,150,numNonActive);

Los argumentos segundo y tercero en la llamada a la función son los números de usuarios activos y no activos, respectivamente. Cualquier combinación puede ser seleccionada mientras que la suma de ambos números sea igual al número total de usuarios. Hay que tener en cuenta que la matriz *userVoteMat* puede contener menos de trescientos usuarios. Esto se debe a que el comando *eachMovieReader* elimina usuarios con muy pocos votos. Por ejemplo, usuarios con un solo voto son completamente inservibles en el ámbito del filtrado colaborativo.

Seguidamente tenemos que dividir el conjunto de ítems de los usuarios activos en vistos y a predecir. El primer grupo es el conjunto de votaciones que el usuario introduce. El sistema es entrenado en base a esta información. El segundo conjunto de ítems se utiliza para evaluar la respuesta del sistema. La recomendación dada para el segundo conjunto de ítems del usuario activo es comparada con la preferencia real, y usando una de las medidas de evaluación se da una calificación numérica. Toda esta distribución de datos se hace dentro del método *EachMovieComparison*, aunque se puede llevar a cabo manualmente si se prefiere.

Para obtener una predicción de la preferencia de un usuario usando filtrado colaborativo basados en memoria usamos el método *predictPreferenceMemBased*. Por ejemplo,

>> predPref=predictPreferenceMemBased(simVec, usersMat, activeAvg)

realiza los métodos de filtrado colaborativo basados en memoria que se describen en [5]. *simVec(j)* contiene la similitud entre el usuario activo y el usuario j, esta similitud debe ser calculada usando algunos de los métodos presentados en [5]. Así mismo, *usersMat* es una matriz sparse con los votos de los usuarios no activos (en las filas) de los diferentes ítems (en las columnas). *activeAvg* es un escalar con la votación media de los usuarios activos.

La predicción obtenida debe ser evaluada usando una de las medidas descritas en el toolkit[1]. Por ejemplo, el comando *rankedEvalCF* evalúa las recomendaciones mediante ranking de acuerdo con los valores de salida media y calcula la utilidad esperada de los ítems ordenándolos en una lista, teniendo en cuenta que la probabilidad de que el elemento sea seleccionado por el usuario decae exponencialmente según este situado más abajo en la lista.

Mediante el siguiente diagrama presentamos esquemáticamente los pasos que hay que llevar a cabo para hacer un sistema de recomendación mediante filtrado colaborativo basado en memoria usando el toolkit "C/Matlab Toolkit for Collaborative Filtering" de Guy Lebanon:

[1] Para más información véase el capitulo 4.2: Medidas de Evaluación

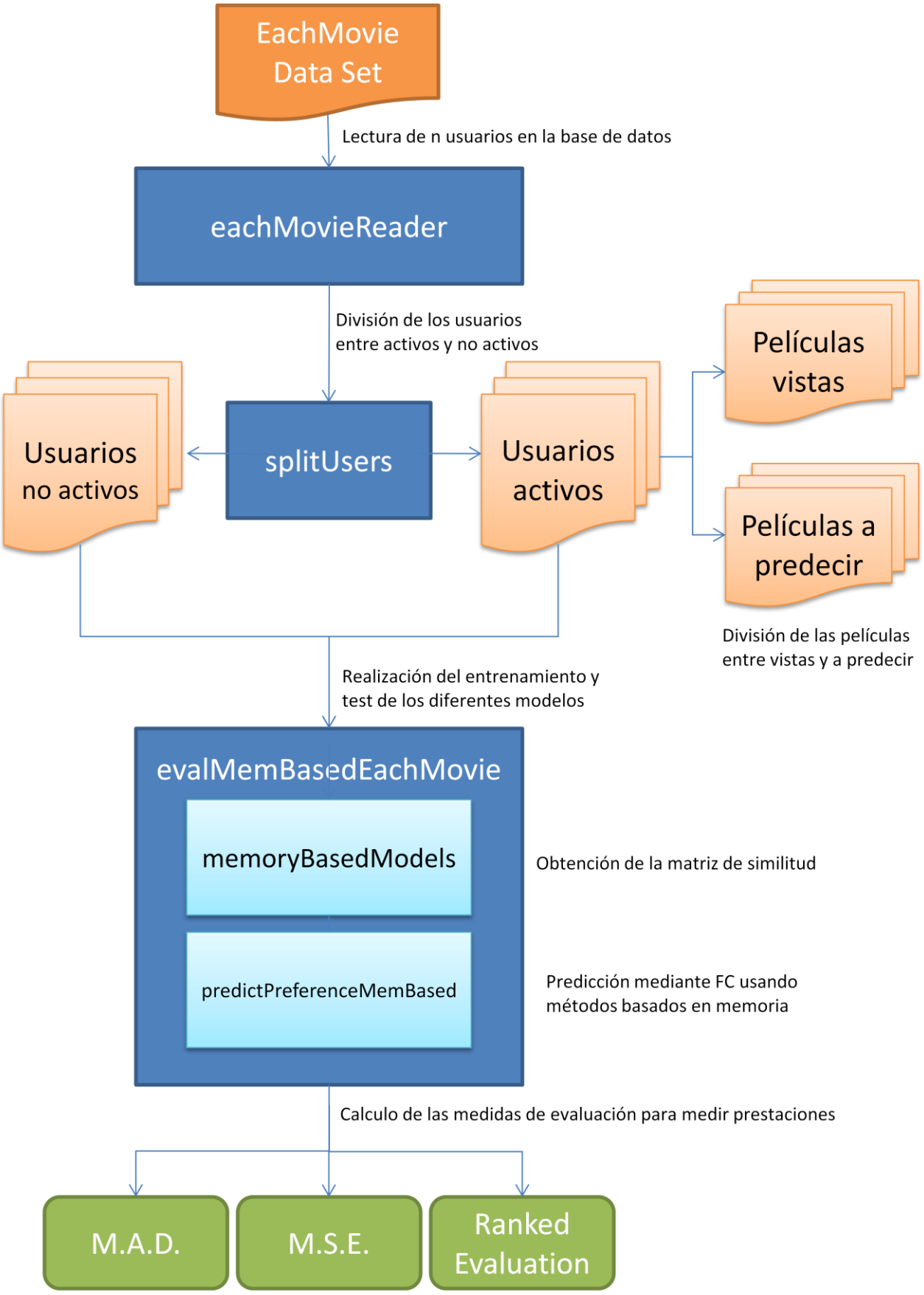

Figura 2.2 Diagrama de una ejecución de un sistema de recomendación con filtrado colaborativo mediante algoritmos basados en memoria usando el toolkit "C/Matlab Toolkit for Collaborative Filtering"

3 DESCRIPCIÓN DE LA PROPUESTA

Este capítulo está dividido en dos apartados. En el primero, haremos una revisión de la propuesta que desarrolla este proyecto, explicando el trabajo y desarrollo realizado.

En la otra parte del capítulo, describiremos algunos algoritmos de agrupamiento que han surgido recientemente y se estudiarán sus propiedades.

3.1 Detalle del trabajo realizado

Antes de hacer la presentación de los métodos de agrupamiento que vamos a introducir en el cálculo de distancias entre usuarios en sistemas de recomendación pasamos a detallar el trabajo experimental llevado a cabo.

El punto de partida a la hora de diseñar la parte experimental es el toolkit para Matlab que desarrolló Guy Lebanon a partir de los artículos *Empirical Analysis of Predictive Algorithms for Collaborative Filtering, Uncertainty in Artificial Intelligence* [5]*, Collaborative Filtering by Personality Diagnosis: A Hybrid Memory and Model-Based approach. Uncertainty in Artificial Intelligence* [8] *y Dependency Networks for Inference, Collaborative Filtering, and Data Visualization* [9].

La idea de este Proyecto Fin de Carrera es construir una alternativa basada en métodos de agrupamiento a los métodos presentes en el toolkit que calculan la matriz de similitud [5] entre usuarios. Como esta nueva funcionalidad no está presente en el software de Guy Lebanon, nuestro proyecto busca evaluar empíricamente si tiene sentido usarla.

El objetivo buscado era el de diseñar una medida de similitud entre usuarios basada en métodos de agrupamiento. La solución propuesta fue la de que los métodos de agrupamiento generaran un espacio de *Prototipos de Usuarios*, proyectar todos los usuarios en ese espacio de *Prototipos de Usuarios* y medir distancias en este nuevo espacio. Esta solución busca la creación de usuarios tipo y la búsqueda de estos a la hora de hacer recomendaciones a los usuarios activos en el sistema.

En primer lugar, creamos un sistema de recomendación basado en métodos de agrupamiento en el cálculo de distancias entre usuarios sencillo para comparar sus resultados con los que

proporcionaban los sistemas de recomendación basados en memoria que implementa Guy Lebanon en su toolkit, es decir, coeficiente de correlación de Pearson y Vector Similarity[2].

En este primer sistema de recomendación se empleó la distancia euclídea en el espacio de las distancias a centroides creado por el algoritmo K-medias. De este modo construimos un sistema de recomendación completo habiendo cambiado la función de similitud entre usuarios, y se podría llevar a cabo un primer experimento para comparar los resultados con los que ofrecían los métodos ya implementados por Guy Lebanon.

A continuación, realizamos más experimentos empleando varias técnicas de agrupamiento más sofisticadas para conseguir diferentes espacios de *Prototipos de Usuarios* y evaluar las ventajas/desventajas comparando las prestaciones de nuestros sistemas de recomendación en diferentes escenarios.

Los algoritmos de agrupamiento que introdujimos fueron Kernel-Kmedias [10] y Mezcla de Gaussianas entrenadas con EM [4].

La introducción de los nuevos métodos de clustering no proporcionaba mejores resultados que K-medias en las medidas de evaluación[3] utilizadas para comparar las prestaciones de los diferentes sistemas de recomendación. No obstante, mediante Kernel K-medias seguíamos teniendo prestaciones similares a los otros dos algoritmos de agrupamiento, pero dependiendo del número de centroides empleado, el tiempo de cómputo era incluso menor que en los sistemas de recomendación mediante filtrado colaborativo basados en memoria.

A la vista de los resultados obtenidos, identificamos un posible problema en los sistemas de recomendación implementados. En las primeras simulaciones parecía que los métodos de agrupamiento estaban construyendo distancias entre usuarios equivalentes, y empeoraban los resultados ofrecidos por los sistemas de recomendación basados en filtrado colaborativo.

El estudio final efectuado consistió en analizar el rendimiento en función del número de usuarios y el número de registros de cada usuario en todos los sistemas de recomendación disponibles para encontrar posibles diferencias en la sensibilidad de los sistemas de recomendación ante diversos escenarios. Para los sistemas de recomendación creados por nosotros que emplean métodos de clustering en el cálculo de distancias entre usuarios, se procede a estudiar las prestaciones de los recomendadores en función del número de centroides utilizado a la hora de crear el agrupamiento.

Por ello, realizamos una serie de simulaciones que en el siguiente capítulo presentamos y discutimos.

[2] Para más información véase el capitulo 2.1.3 Medidas de Distancia o Similitud
[3] Ver capítulo 4, apartado 2: Medidas de Evaluación

3.2 Métodos Clustering

El agrupamiento es uno de los problemas más estudiados en aprendizaje no supervisado. Los algoritmos de este tipo buscan agrupar en distintos conjuntos datos que tienen características similares. Un grupo es una recopilación de objetos que son similares entre ellos y diferentes respecto a los miembros de otros clúster.

En este apartado haremos una breve descripción de algunas técnicas de agrupación que hemos introducido dentro de nuestros experimentos.

3.2.1 K-medias

El algoritmo K-medias o K-means es una de las técnicas de agrupamiento más utilizadas en el estado del arte.

En estadística y aprendizaje máquina, el algoritmo de agrupamiento K-means es un método de búsqueda de particiones que tiene por objeto realizar particiones de un número N observaciones en un número K de grupos en los que cada observación pertenece al grupo con la media más cercana. Es similar al algoritmo de mezcla de Gaussianas mediante EM[4] ya que en ambos se intenta encontrar los centros de agrupaciones naturales en los datos, así como que en ambos algoritmos emplean una aproximación iterativa para lograr el refinamiento.

Dado un conjunto de datos $x_i \in X, i = 1,2,\ldots,N$ y el número de grupos a recuperar, K, su objetivo es encontrar las *K* particiones, conjuntos o grupos $S = \{S_1, S_2, \ldots, S_K\}$ a fin de minimizar la suma de los cuadrados de los datos dentro del grupo:

$$\arg\min_{S} \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in S_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2,$$

Donde $\mu_i = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$ es la media del grupo S_j, siendo $|S_j|$ el número de elementos del grupo (1)

Aunque este problema es NP-completo, una buena solución puede obtenerse, generalmente, usando una técnica de refinamiento iterativo conocida a menudo como algoritmo de Lloyd o simplemente "algoritmo *K-means*".

A continuación se presenta una tabla donde se muestran los principales pasos a seguir para el algoritmo iterativo K-means:

[4] Para más información ver apartado 3.2.1

Algoritmo de Agrupamiento K-means

Dados los datos $x_1, \ldots, x_N \in X$ y el número de grupos (clusters) a recuperar, *K*.

inicializar

Escoger aleatoriamente $\mu_j \in X$, para $j = 1,2,\ldots,K$

repetir

Asignar el dato x_i a la partición S_j con el centro más cercano μ_j , para $i = 1,2,\ldots,N$

Recalcular los μ_j ,el centro del *cluster* S_j, para $j = 1,2,\ldots,K$ usando (1)

hasta no existan cambios en las asignaciones.

Figura 3.1 Pseudocódigo del algoritmo K-medias

Este tipo de algoritmos de agrupamiento presentan algunos inconvenientes o limitaciones. La primera es que la convergencia en el óptimo global no está garantizada. Para problemas con muchos ejemplares requiere de un gran número de iteraciones, e incluso después de múltiples ejecuciones no son capaces de obtener resultados que sean mejores. Este hecho hace que el algoritmo sea extremadamente sensible a la elección inicial de los centroides. La segunda limitación del algoritmo K-means relatada en la literatura es su incapacidad para agrupar conjuntos de datos que no son linealmente separables.

Con respecto a la primera de las limitaciones, generalmente es necesaria una elección más sólida de los centroides iniciales y numerosas investigaciones se han centrado en solucionar este problema. Por ejemplo, en [11] se propone un algoritmo híbrido, que combina los algoritmos de K-medias y de Particle Swarm Optimization (PSO) [12]. El algoritmo PSO realiza una búsqueda global en el espacio de soluciones y el enfoque híbrido se utiliza para encontrar los centroides iniciales para algoritmo K-means. Una estrategia similar con este mismo propósito es la propuesta por [13], donde el algoritmo híbrido reúne los algoritmos K-means y los evolución del diferencial (DE) [14]. En cuanto a la segunda limitación existen diversas estrategias que intentan la adaptación del algoritmo de agrupamiento para conjuntos en los que los datos no son linealmente separables, como pueden ser el algoritmo Kernel K-medias, los algoritmos de agrupamiento espectral y los algoritmos de partición de grafos.

No obstante y pese a las limitaciones que presenta, esta técnica de agrupamiento gracias a su baja complejidad computacional, es muy utilizada ya que en la práctica es a menudo suficiente para elegir el mejor resultado de la agrupación de un conjunto de datos, realizándose aleatoriamente la elección inicial de los centroides.

3.2.2 Kernel K-medias

El algoritmo de agrupamiento Kernel K-means [15], [16] aparece en un intento de resolver la limitación del método K-means en relación de la separabilidad lineal. El algoritmo Kernel K-medias intenta controlar esta limitación proyectando el conjunto de características de los datos a un espacio con un número de dimensiones mayor mediante una función $\phi : R^m \rightarrow R^q$, con $q > m$, donde los grupos de este conjunto son linealmente separables [17].

El algoritmo Kernel K-means utiliza una función ϕ de manera alternativa al mapeo del conjunto de datos. Al igual que el algoritmo de k-medias, este algoritmo busca aquellos grupos para los que la función objetivo

$$f(x) = \sum_{v=1}^{k} \sum_{x_i \in C_v} \|\phi(x_i) - m_v\|^2$$

se minimiza, en el caso de que la medida de similitud es la distancia euclídea. El valor del centroide m_v se expresa por:

$$m_v = \frac{1}{n_v} \sum_{x_i \in C_v} \phi(x_i)$$

Se define un kernel $K : \Re^q x \Re^q \rightarrow R$, de modo que $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$, se demuestra que:

$$f(x) = \sum_{v=1}^{k} \sum_{x_i \in C_v} \|\phi(x_i) - m_v\|^2 =$$

$$= \sum_{v=1}^{k} \sum_{x_i \in C_v} \left[K_{ii} - \frac{2}{|n_v|} \sum_{x_j \in C_v} K_{ij} + \frac{1}{|n_v|^2} \sum_{x_j \in C_v} \sum_{x_l \in C_v} K_{jl} \right]$$

Algoritmo de Agrupamiento Kernel K-means

Dados los datos $x_1, \ldots, x_N \in X$ y el número de grupos (clusters) a recuperar, C.

inicializar

Escoger aleatoriamente los centroides $m_v \in X$, para $j = 1,2,\ldots,C$

repetir

Calcular la distancia euclídea de cada dato x_i a los centroides en el espacio transformado usando el kernel K y asignar el dato x_i al cluster con el centroide más cercano,

$$j^*(x_i) = argmin_j \|\phi(x_i) - m_v\|^2$$

Obtener los clusters actualizados,

$$C_j = \{ x_i \ : \ j^*(x_i) = j \}$$

hasta no existan cambios en las asignaciones.

Figura 3.2 Pseudocódigo del algoritmo Kernel K-medias

3.2.2.1 *Kernel K-medias Ponderado*

En [18] se propone una variación del algoritmo Kernel K-means, llamado Kernel K-means Ponderado. En este algoritmo se introduce un peso w_i para cada dato x_i. Por tanto tendríamos la siguiente función objetivo:

$$f(x) = \sum_{v=1}^{k} \sum_{x_i \in C_v} w_i \|\phi(x_i) - m_v\|^2$$

que debe ser minimizada, y el centroide m_v se expresa por:

$$m_v = \frac{\sum_{x_i \in C_v} w_i \phi(x_i)}{\sum_{x_i \in C_v} w_i}$$

De manera semejante se tiene:

$$f(x) = \sum_{v=1}^{k} \sum_{x_i \in C_v} w_i ||\phi(x_i) - m_v||^2 =$$

$$= \sum_{v=1}^{k} \sum_{x_i \in C_v} w_i \left[K_{ii} - \frac{2 \sum_{x_j \in C_v} w_j K_{ij}}{\sum_{x_j \in C_v} w_j} + \frac{\sum_{x_j \in C_v} \sum_{x_l \in C_v} w_j w_l K_{jl}}{[\sum_{x_j \in C_v} w_j]^2} \right]$$

En el algoritmo Kernel K-means ponderado, si todos los pesos son unitarios, éste se reduce al algoritmo Kernel K-means, y además si la función ϕ es la función identidad se reduce al algoritmo K-means.

Hay que tener en cuenta que aunque con estos dos algoritmos se resuelve el problema que presentaba el K-means original de no poder trabajar con conjuntos de datos linealmente no separables, siguen presentando la misma limitación observada anteriormente de la convergencia a mínimos locales. En [16] se hace una propuesta para resolver el problema de la falta de globalidad de las soluciones generadas por los métodos descritos (se alcanzan mínimos locales).

3.2.3 Mezcla de Gaussianas entrenadas con EM

Los modelos de mezclas de gaussianas se diseñaron para trabajar dentro del ámbito de las técnicas de agrupamiento, aunque también está muy extendido su uso en problemas de estimación de densidades de distribuciones.

El modelado mediante mezcla de distribuciones es una teoría muy útil para la descripción de sistemas multimodales. La potente descripción de los datos y señales conseguida con este modelado mediante mezclas, ha hecho que este método sea aplicado en diferentes disciplinas, como la bioinformática, ingeniería, robótica y economía entre otras.

El modelo de mezclas más ampliamente estudiado y que además, ha demostrado un buen funcionamiento en múltiples aplicaciones, es el modelo de mezcla de Gaussianas [19]. Estos modelos poseen una forma lo suficientemente flexible para dar cabida a gran variedad de distribuciones y poseen también la ventaja de proporcionar una visión sobre el mecanismo que genera los datos, así como una interpretación racional de los diferentes parámetros. La interpretación de las GMMs es que los datos están generados por K distribuciones gaussianas, cuyos tamaños son proporcionales a su peso en la mezcla

Para dicho modelo, los parámetros desconocidos son los pesos w_j, las medias y las varianzas. La función de densidad de un conjunto de datos dado corresponde a un GMM con un número K componentes se puede expresar como

$$p(y|\theta,z) = \sum_{j=1}^{K} w_j \frac{1}{\sqrt{2\pi}\sigma_j} exp\left(-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right).$$

El modelado mediante mezclas trata de, a partir de un número de muestras u observaciones, realizar la estimación de los parámetros de cada una de las distribuciones y subpoblaciones que componen la mezcla.

El algoritmo *Expectation Maximization* (EM) es la herramienta generalmente más utilizada para estimar los parámetros de un GMM, $\theta = \{\mu_j, \sigma_j^2, w_j, \sum_j \text{ con } j = 1,2,\dots,K\}$. El algoritmo EM tiene como objetivo la maximización de la función de densidad de los datos en función de los parámetros a estimar θ, lo que se llama verosimilitud.

El algoritmo EM es un proceso iterativo. En cada iteración, se utilizan las estimaciones actuales de los parámetros para calcular un indicador de cómo cada gaussiana "explica" cada observación (Paso E). Después, se utilizan estos indicadores para encontrar las estimaciones actualizadas de los parámetros θ (Paso M). A continuación se presenta el algoritmo EM estándar:

Algoritmo de Agrupamiento Mezcla de Gaussianas entrenadas con EM

Inicializar θ^0 y $t = 0$;
repetir
paso E: Calcular $Q\left(\theta^0, \theta^t\right) = E\left[\log p\left(\mathbf{x}^g, \mathbf{x}^m | \theta | \mathbf{x}^g, \theta^t\right)\right]$;
paso M: $\theta = argmax_\theta Q\left(\theta, \theta^t\right)$;
t = t+1;
hasta Se satisfaga la condición de convergencia
Salida: θ.

Figura 3.3 Pseudocódigo del algoritmo Mezcla de Gaussianas entrenadas con EM

A pesar de su popularidad para estimar las GMMs, el algoritmo EM presenta una serie de deficiencias cuando se aplica a grandes bases de datos. EM garantiza la convergencia a un máximo de la función de verosimilitud, sin embargo, dependiendo del punto de partida, este máximo puede corresponder a uno local en lugar del máximo nivel global. Para resolver este problema, el algoritmo normalmente se ejecuta en varias ocasiones, utilizando diferentes puntos de partida. Esta solución es, sin embargo, inviable cuando se trata de grandes bases de datos de alta dimensión, debido a la considerable carga computacional del proceso.

Otro de los problemas del algoritmo EM cuando es aplicado a grandes bases de datos surge cuando se tiene la intención de realizar la selección del modelo. El algoritmo EM asume que el número de distribuciones gaussianas que participan en la mezcla es fijo, por lo que este número no está incluido en el conjunto de parámetros que se actualizará durante iteraciones. Los enfoques tradicionales estiman el número de gaussianas mediante el número de veces que el algoritmo es ejecutado, utilizando diferente número de gaussianas definidas a priori. Los modelos se comparan mediante la calidad de indicador.

4 VALIDACIÓN EXPERIMENTAL

En el capítulo cuarto de este proyecto vamos a presentar el estudio experimental comparativo realizado, donde mostraremos los experimentos simulados y sus resultados.

En el primer apartado comenzaremos describiendo la base de datos que ha sido usada. Posteriormente se detallan las medidas de evaluación con las que van a hacerse las comparaciones entres sistemas de recomendación, y finalmente se exponen las configuraciones de los escenarios con los que se han ejecutado los experimentos propuestos, su discusión y resultados.

4.1 Base de Datos

Para trabajar con nuestros experimentos hemos empleado la base de datos EachMovie[5] de GroupLens.

HP/Compaq Research creó el sistema de recomendación de películas EachMovie. Cuando EachMovie fue discontinuada en 2004 quedo disponible para el uso en investigación. La base de datos actual de películas de GroupLens, MovieLens, estuvo originalmente basada en EachMovie.

Contiene 2.811.983 votos ingresados por 72.916 usuarios para 1628 películas diferentes donde la puntuación para cada película es de 1 a 5. Esta base de datos ha sido usada en numerosas publicaciones sobre filtrado colaborativo.

[5] Disponible en GroupLens/Data Sets - http://www.grouplens.org/node/12

4.2 Medidas de Evaluación

Cuando se presenta un nuevo algoritmo de filtrado colaborativo es necesario medirlo en base a alguna métrica para saber qué tan exactas resultan sus predicciones. Es necesario escoger unas métricas para cuantificar la diferencia entre un estimador y el verdadero valor de la cantidad que se estima para evaluar el rendimiento de los algoritmos propuestos y así poder de esta manera comparar sus prestaciones.

A continuación, se describen algunas medidas de evaluación que se utilizan comúnmente en el área de la clasificación. Estas medidas nos darán información acerca de cuan efectivos son los sistemas utilizados.

La evaluación de un sistema de recomendación se basa, comúnmente, en el cálculo de la matriz de confusión. Esta matriz nos da información sobre el comportamiento del sistema de clasificación y como éste se "confunde" en sus predicciones.

4.2.1 Mean Absolute Deviation

La métrica más utilizada y aceptada en el área para evaluar la precisión de los algoritmos de filtrado colaborativo es la MAD, que mide la desviación media entre el valor de predicción de un ítem y el valor de puntuación (rating) real que asigna el usuario al ítem.

$$D_m = \frac{1}{N}\sum_{i=1}^{n}|x_i - \overline{x}|$$

Utilizando esta métrica, cada elemento del conjunto de prueba es tratado de la misma manera. Esto significa que se le da el mismo peso al error cometido en calcular la predicción de un elemento donde su valor real está por encima o por debajo del promedio de la escala de valores.

Por ejemplo, si la escala es de 1 a 5, el error en un ítem que tiene una puntuación de un punto tiene el mismo valor que el error que se comete en un elemento con una puntuación de tres o cinco puntos. Esencialmente, el error medio absoluto proporciona el mismo peso a los errores sea cual sea el elemento considerado. Esto hace que el error medio absoluto sea el más apropiado para los sistemas que no hacen consideraciones o separaciones de los ítems basados en algo que no sea las puntuaciones disponibles.

Esta métrica tiene la ventaja de ser muy simple de calcular y los resultados son fácilmente interpretables.

4.2.2 Mean Square Error

Estas variaciones apuntan a dar mayor importancia a los errores más grandes. Por ejemplo, si se utiliza el MSE, se estaría amplificando el efecto de los mayores errores, ya que un error de valor 1, suma 1 al total, mientras que un error de 2 sumaría 4.

El MSE es una función de riesgo, que corresponde al valor esperado de la pérdida de error cuadrado o pérdida cuadrática ya que mide el promedio del cuadrado de la cantidad en que el estimador difiere de la cantidad que se estima. Esta diferencia entre ambas cantidades se puede producir debido a la aleatoriedad, o porque el estimador no tiene en cuenta alguna información que podría dar lugar una estimación más precisa.

Matemáticamente el MSE se puede interpretar como el segundo momento del error, por lo que incorpora información tanto acerca de la varianza del estimador como de su sesgo, coincidiendo que para un un estimador insesgado el MSE es la propia varianza. Al igual que la varianza, el MSE tiene la misma unidad de medida que el cuadrado de la cantidad que se estima. En comparación con la desviación estándar, si se calcula la raíz cuadrada de la MSE se obtiene la raíz cuadrada del error cuadrático medio o también llamada desviación típica. Esta medida tiene las mismas unidades que la cantidad que se estima, y en el caso de tener un un estimador insesgado, el RMSD es directamente la raíz cuadrada de la varianza también conocido como el error estándar.

El MSE de un estimador $\hat{\theta}$ con respecto al parámetro estimado θ se define como:

$$\mathrm{MSE}(\hat{\theta}) = \mathrm{E}\left[(\hat{\theta} - \theta)^2\right]$$

El MSE lo que evalúa es la calidad de un estimador en términos de variación y sesgo.

Dado que el MSE es una esperanza matemática, su valor será un escalar. Puede resultar ser una función del parámetro desconocido θ, pero no dependerá de cantidades aleatorias. Sin embargo, cuando el MSE se calcula para un estimador particular de θ cuyo valor real no es conocido, estará sujeto a un error de estimación. Esto significa que hay casos en los que se puede tratar como una variable aleatoria.

4.2.3 Ranked Evaluation

Esta medida de evaluación es una métrica propia de los sistemas de recomendación. Mide la preferencia real que se espera del ítem elegido cuando la probabilidad de elegir un tema recomendado decae exponencialmente con su ubicación en una lista ordenada de las recomendaciones.

Es una generalización de la que aparece en [9] para valores de preferencia múltiple. En este estudio los autores siguen un criterio que trata de medir la utilidad que una lista de

recomendaciones tiene para usuario. Por supuesto, diferentes usuarios tendrán también diferentes funciones de utilidad. Esta medida ofrece una buena aproximación a través de muchos usuarios.

El escenario que utilizan en el estudio es uno donde a un usuario se le muestra una lista ordenada de elementos y, a continuación analiza la lista de temas preferidos empezando por la parte superior. En algún momento, el usuario deberá dejar de mirar a más artículos. Entonces, *p(k)* es la probabilidad de que un usuario examine el elemento k-ésimo de una lista de recomendaciones antes de detener su exploración, donde la primera posición viene dada por k= 0. Entonces, un criterio razonable es:

$$cf_{accuracy_1}(list) = \sum_k p(k)\delta_k$$

donde δ_k es 1 si el elemento k es preferido y 0 en caso contrario. Se puede asumir que p(k) es una función exponencialmente decreciente tal como :

$$p(k) = 2^{-k/a}$$

donde a es la posición donde el elemento será visto con probabilidad del 50%.

4.3 Escenarios

Se ha realizado una simulación usando una base de datos de 1000 usuarios en total. Presentaremos diferentes escenarios en los que evaluaremos las prestaciones de nuestra propuesta frente a las obtenidas mediante los sistemas de recomendación para filtrado colaborativo basado en memoria que aparecen actualmente en la literatura.

Vamos a comparar el rendimiento de los diferentes escenarios utilizando las medidas de evaluación anteriormente expuestas, a saber, mean absolute deviation (MAD), mean square error (MSE) y ranked evaluation (RE).

Las simulaciones que se han realizado eligiendo del total de usuarios de la base de datos escogida un número aleatorio de usuarios suficientemente representativo. Vimos que con un total de 1000 usuarios en el sistema ya conseguíamos resultados consistentes y que aumentar este número no hacía más que aumentar la carga computacional de las simulaciones.

Se escogió usar un 25%, un 50%, un 75% y un 90% de usuarios activos en el sistema, que son aquellos a los que les queremos recomendar contenido. Esta variable es la llamada *activePerc* en las gráficas del siguiente apartado.

Así mismo del total de películas calificadas por cada usuario activo se separa un tanto por ciento de ellas para usarlas como películas vistas, y el tanto por ciento restantes serán las películas a predecir. Comparando estas predicciones con la calificación real podremos evaluar la salida de nuestro sistema de recomendación. El ratio entre películas que ya han sido vistas y el número de películas a recomendar lo hemos variado en un 25%, 50%, 75% y 90%. Esta será la variable *percPredicted*.

En los experimentos se cambiará el número de centroides para los métodos de agrupamiento en un 5%, 10%, 20% y 30% del total de usuarios para ver la influencia de utilizar un mayor o menor número de clusters e intentar encontrar el número óptimo de grupos con el que se consiga un buen compromiso entre resultados obtenidos y carga computacional.

Por último, mostramos los parámetros de configuración de los sistemas de recomendación empleados. Éstos han sido fijados mediante simulaciones buscando obtener a la salida del sistema de recomendación las mejores estimaciones, siempre dentro de los márgenes que recomendaban los creadores de los algoritmos.

4.3.1 K-medias ponderado

El script usado para la ejecución de este algoritmo fue programado durante la práctica final de la asignatura de Tratamiento Digital de Señales durante el curso académico 2007/2008[6].

La inicialización de los centroides se realiza de manera aleatoria eligiéndolos entre los usuarios presentes en el sistema. El número máximo de iteraciones del algoritmo ha sido fijado a 100, o hasta los centroides no cambien de una iteración a la siguiente.

Para el cálculo de la distancia de cada dato a los centroides se ha empleado la distancia euclídea.

4.3.2 Mezcla de Gaussianas entrenadas con EM

El algoritmo usado para simular este método ha sido codificado por Ian T. Nabney basándose en las técnicas descritas en *Neural Networks for Pattern Recognition* [4]

En primer lugar se crea un modelo de mezclas de gaussianas usando para la covarianza una estructura esférica. Después inicializamos los parámetros usando los usuarios activos en 5 iteraciones como máximo. Finalmente aplicamos el algoritmo EM con un máximo de 30 iteraciones o una precisión absoluta en el error de 10^{-4} como condición de parada.

4.3.3 Kernel K-medias

El toolbox de Stijn Vanderlooy ha sido el empleado para simular este algoritmo de agrupamiento [10].

La configuración elegida ha sido usando un kernel del tipo polinómico iterando 100 veces.

[6] Matlab trae de serie implementado su propio algoritmo K-medias. Para más información se puede consultar el manual de ayuda al usuario o usando el comando help kmeans

4.4 Presentación y discusión de resultados

En las siguientes figuras podemos observar el comportamiento de la medida *mean absolute desviation, mean absolute deviation* y *ranked evaluation* para todos los sistemas de recomendación utilizando para realizar el filtrado colaborativo tanto los métodos existentes basados en memoria como los algoritmos de agrupamiento propuestos en el proyecto.

Para que los datos obtenidos tengan consistencia estadística las gráficas presentan el promediado 10 simulaciones. Representamos tanto la media de las simulaciones como la varianza en torno a la media para ver la dispersión de los resultados.

4.4.1 Mean Absolute Deviation

Primero empezamos comentando los resultados que han sido obtenidos para la desviación media absoluta.

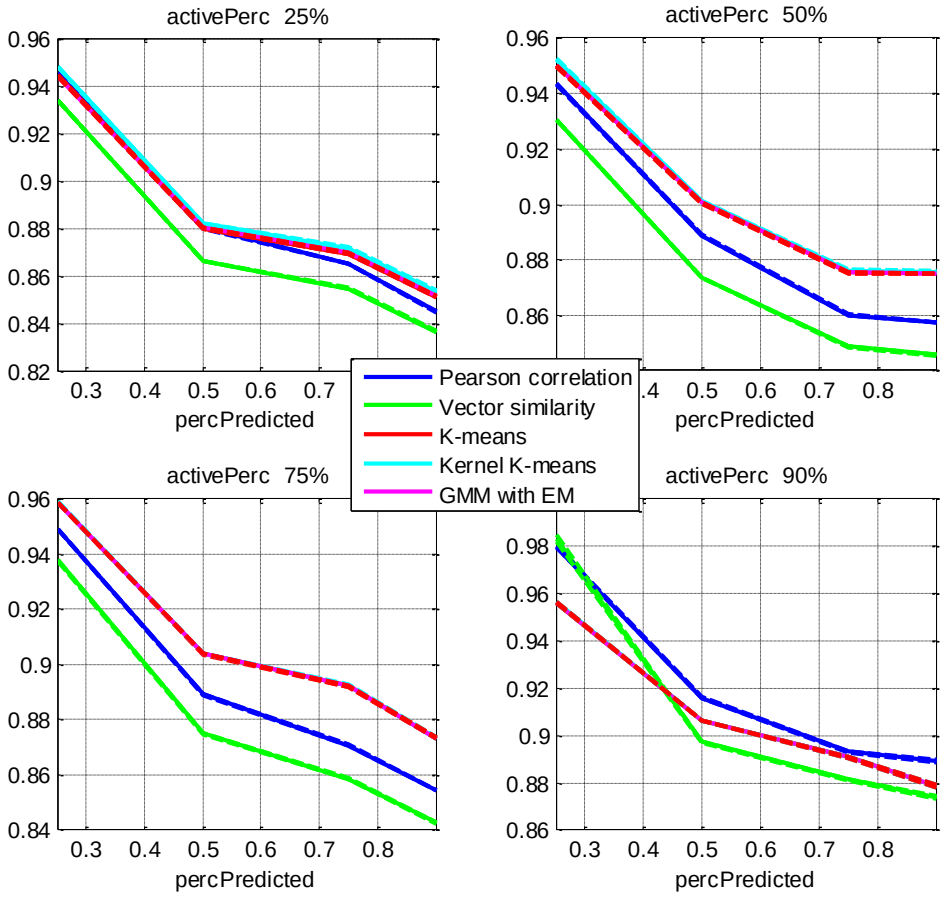

Figura 4.1 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 5% del total de usuarios como centroides

Como podemos ver en la figura 4.1, en líneas generales cuanto más aumente el número de usuarios con películas votadas y el número de películas vista por los usuarios activos los sistemas de recomendación propuestos mejoran sus predicciones. En algunos métodos se ve que cuando el número de contenido visionado es muy elevado las prestaciones empeoran. Esto puede ser debido a la dificultad de encontrar películas que aún no han sido vistas y a usuarios con gustos afines que sí las hayan visto y sirvan para hacer la recomendación, por lo que acertar en la predicción se hace más complicado.

Para el conjunto simulado podemos ver que en la mayoría de los casos los métodos de agrupamiento no llegan a alcanzar los resultados obtenidos por los sistemas de recomendación mediante filtrado colaborativo basados en memoria.

Cuando el número de usuarios activos en el sistema con respecto al total es del 25% vemos que obtenemos resultados equivalentes usando clustering y Pearson correlation coefficient para una relación de películas vistas/predichas por los usuarios activos inferior al 50%. Para un 90% de usuarios activos los algoritmos de agrupamiento mejoran en todo caso al coeficiente de correlación de Pearson, y en el caso particular de una proporción de películas vistas/predichas por los usuarios activos del 25% también mejoran el rendimiento de los sistemas de recomendación mediante filtrado colaborativo basados en memoria usando vector similarity.

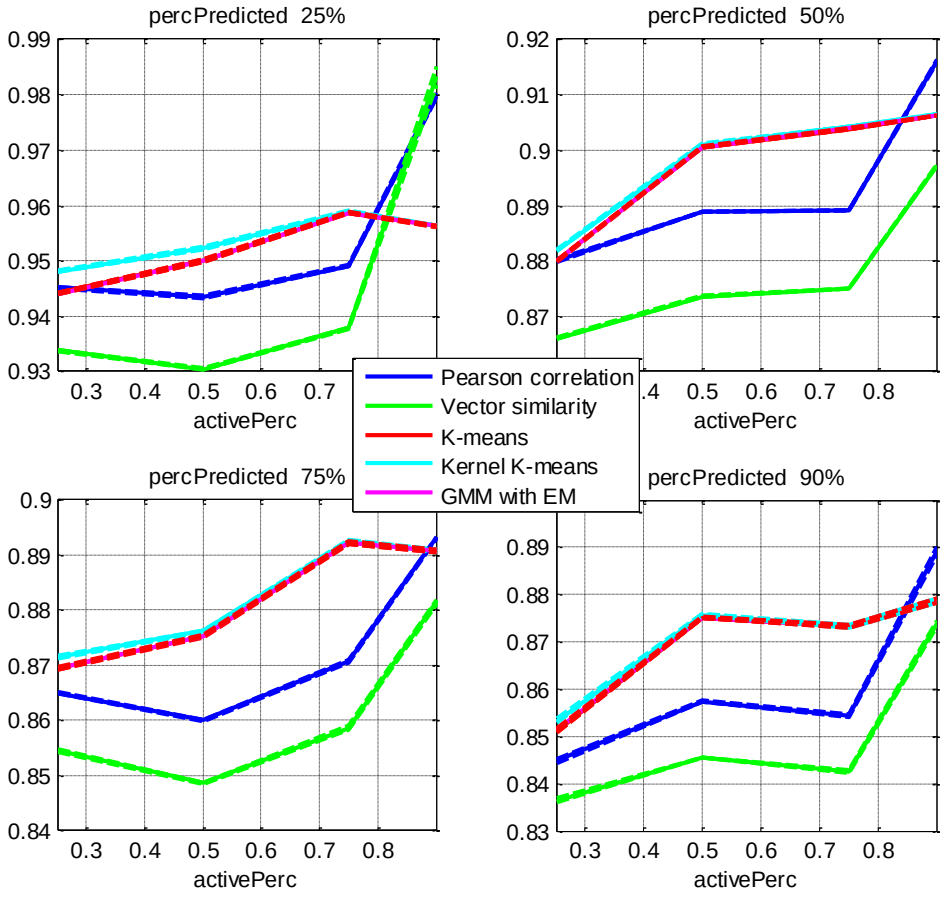

Figura 4.2 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 5% del total de usuarios como centroides

La figura 4.2 presenta la misma simulación que se hizo en la figura anterior pero ahora desde el punto de vista opuesto. Así podemos ver mejor cómo se comportan los métodos usados a medida que aumenta la cantidad de usuarios activos en el sistema manteniendo fijo el número de películas vistas por los usuarios activos. Esto puede ser útil para ver cuál es la cantidad adecuada de usuarios que tendría que tener nuestra base de datos para empezar a dar buenas predicciones en función de cuanto les guste a los usuarios del sistema opinar y votar las películas vistas, ya que no es lo mismo tener un sistema con usuarios que contribuyan con muchos aportes que con pocos.

En todos los sistemas de recomendación no es posible dar resultados a los usuarios que han visto muy pocas películas. Este es un problema latente en los sistemas de recomendación que tienen gran dificultad en hacer recomendaciones a usuarios nuevos en el sistema con poco contenido calificado.

El resto de simulaciones que presenten los mismos resultados irán una a continuación de otra y serán comentadas conjuntamente para mayor comodidad a la hora de interpretar los resultados matemáticos obtenidos.

En las figuras 4.3 y 4.4 pueden verse ahora el comportamiento habiendo incrementado al 10% del total de usuarios el número de centroides.

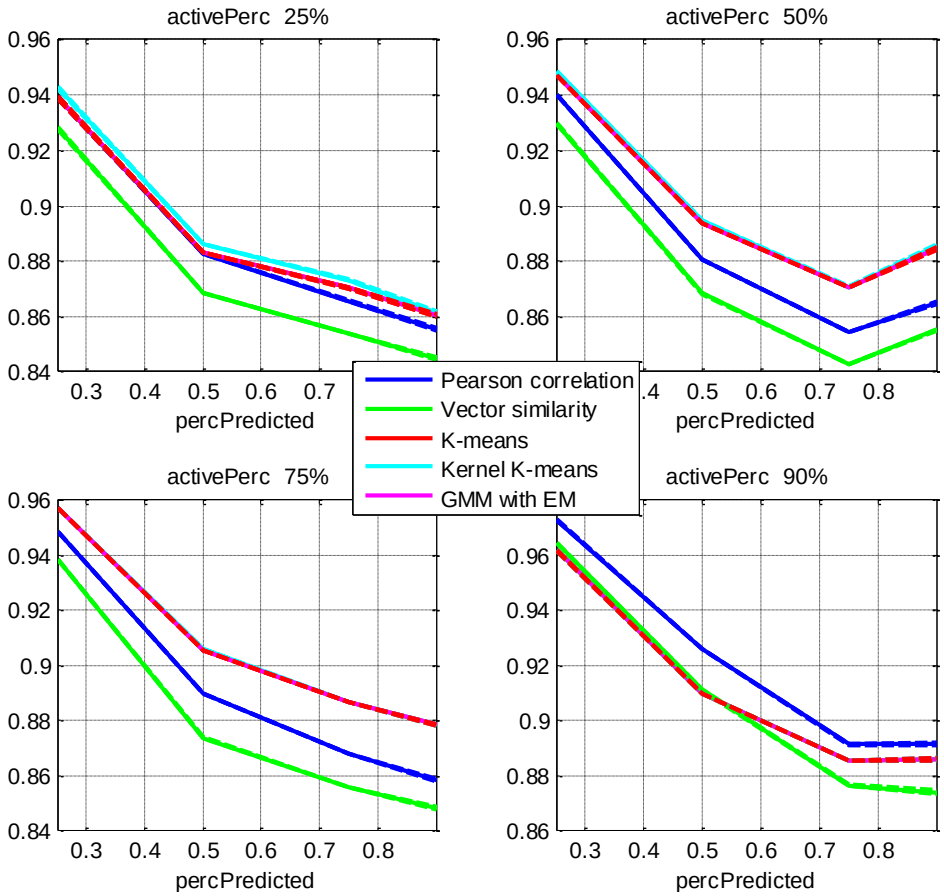

Figura 4.3 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 10% del total de usuarios como centroides

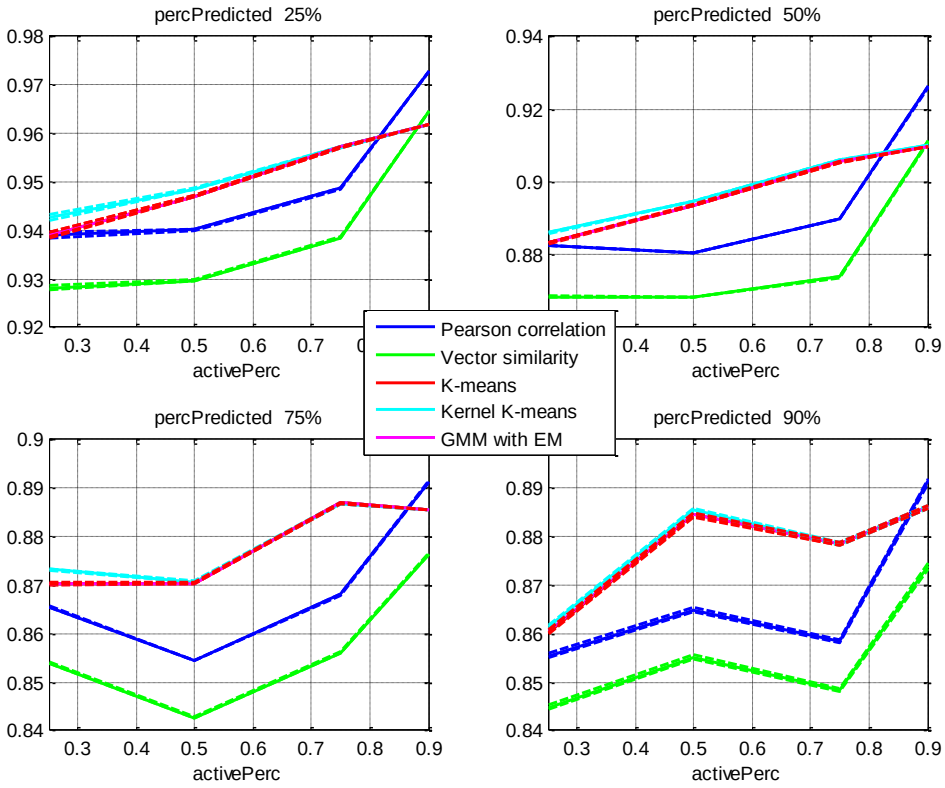

Figura 4.4 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 10% del total de usuarios como centroides

Al haber doblado el número de centroides desde la simulación anterior a esta, podemos decir que usar clustering para el cálculo de la matriz de similitud empieza a dar resultados cercanos a los proporcionados en los sistemas de recomendación mediante filtrado colaborativo basados en memoria.

Vemos que para un número de usuarios activos en el sistema elevado, 90%, los métodos de agrupamiento son mejores que usar el coeficiente de correlación de Pearson en todo caso y que vector similarity para casos puntuales.

En las figuras 4.5 y 4.6 pueden verse ahora el comportamiento habiendo fijado el número de centroides al 20% del total de usuarios del sistema.

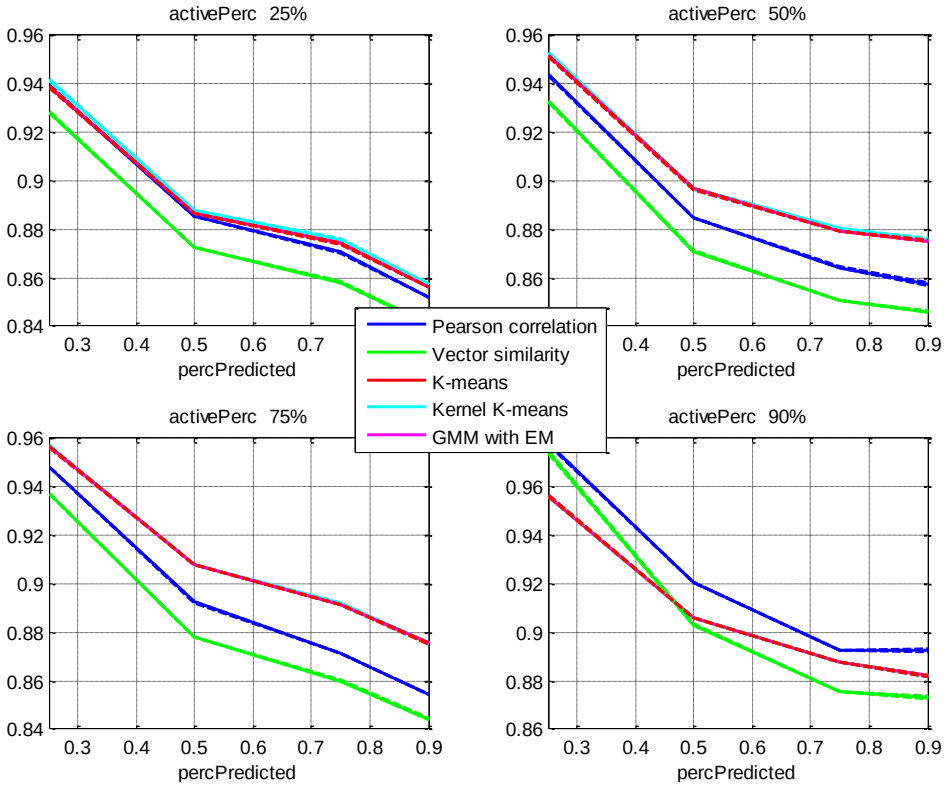

Figura 4.5 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 20% del total de usuarios como centroides

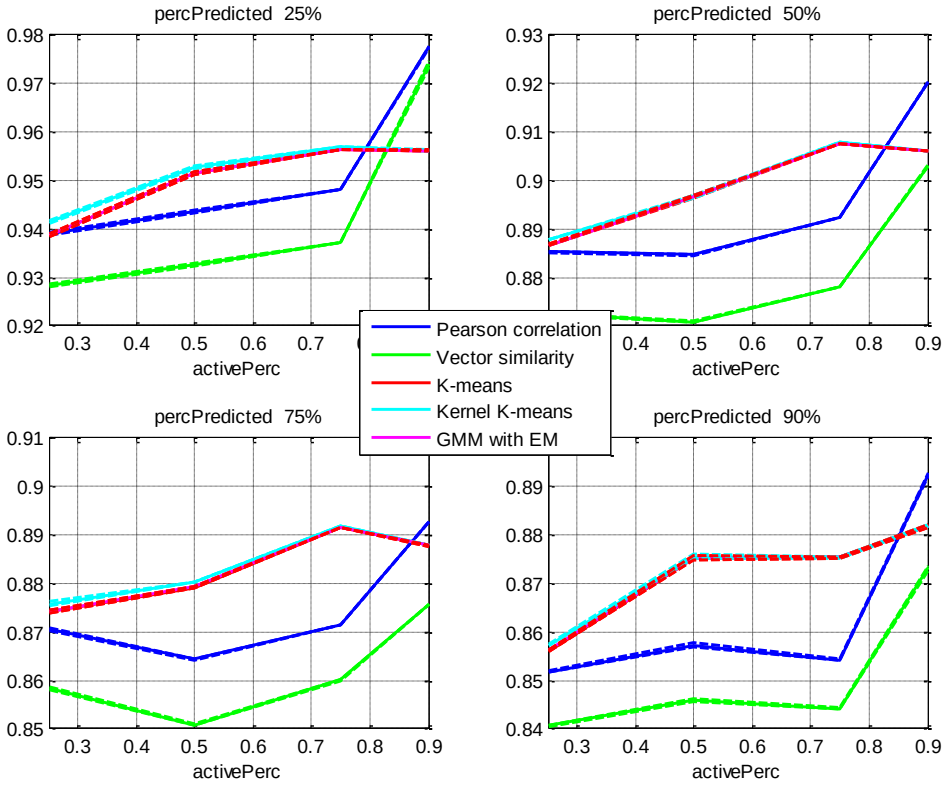

Figura 4.6 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 20% del total de usuarios como centroides

Aumentar a 20% del total de usuarios el número de centroides no hace que la situación cambie demasiado. Todos los algoritmos de agrupamiento empleados empiezan a dar soluciones equivalentes, siendo el algoritmo de Kernel K-means el que peor resultado da para un número menor de usuarios activos, por lo que podemos pensar que el número de centroides empleados es ya suficientemente grande.

Como en el caso anterior para cuando hay muchos usuarios activos se consiguen mejoras en algunos casos frente a Vector Similarity y resultados iguales o superiores frente a usar el coeficiente de correlación de Pearson.

En las figuras 4.7 y 4.8 pueden verse ahora el comportamiento habiendo fijado el número de centroides al máximo simulado, es decir, al 30% del total de usuarios del sistema.

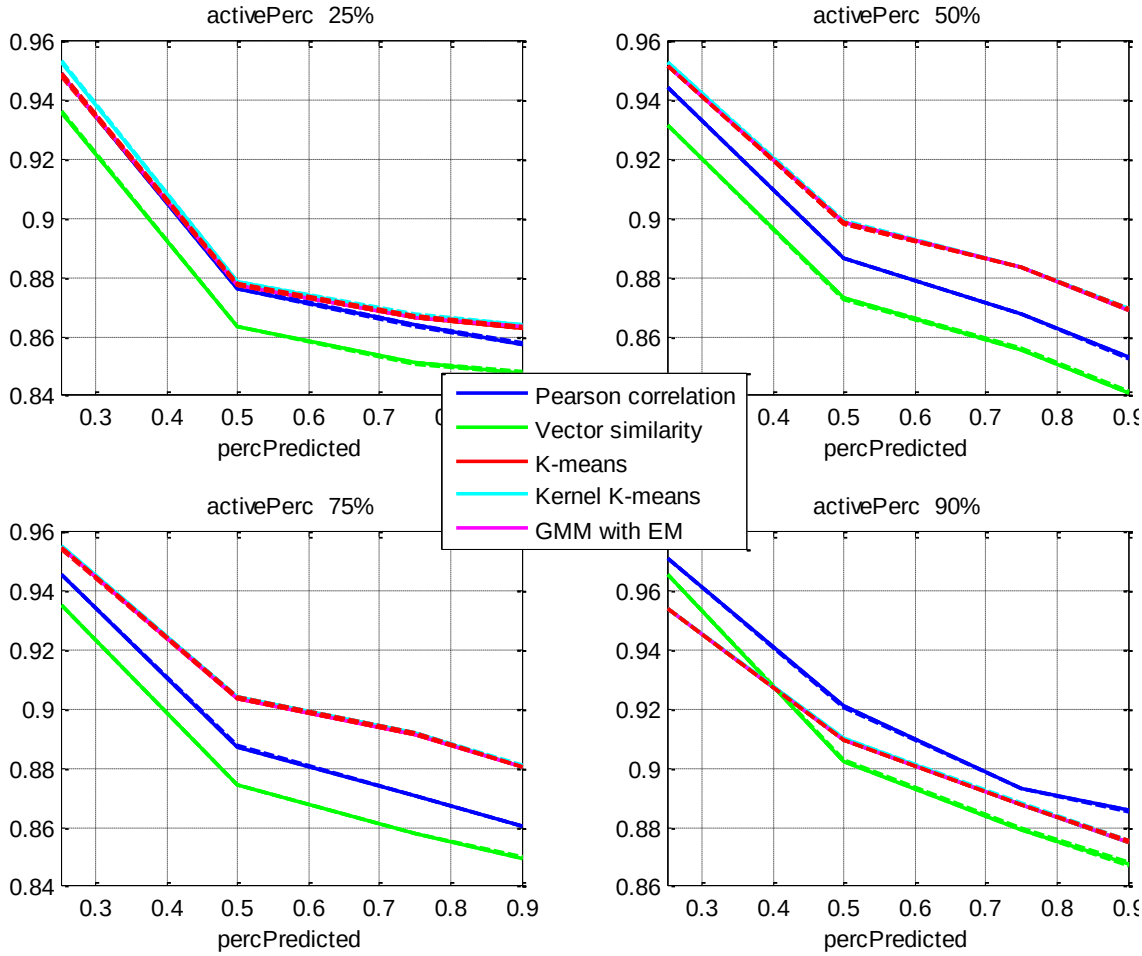

Figura 4.7 Mean absolute deviation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 30% del total de usuarios como centroides

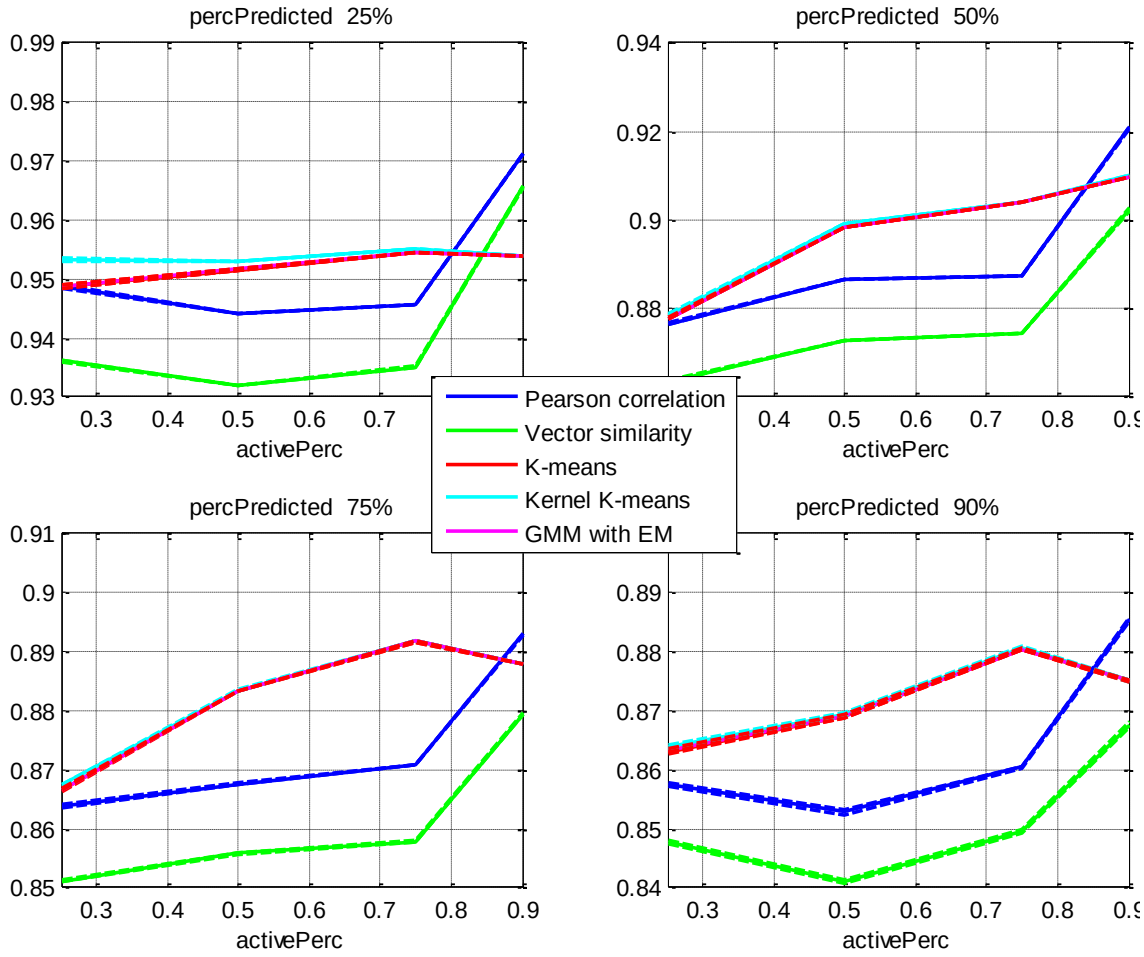

Figura 4.8 Mean absolute deviation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 30% del total de usuarios como centroides

Finalmente para un total de 300 centroides todos los algoritmos de agrupamiento empleados convergen a la misma solución y empeoran el resultado de los sistemas basados en memoria, salvo como en ocasiones anteriores, para un porcentaje alto de usuarios activos en el sistema.

Ahora se va a pasar a evaluar individualmente cada método en los escenarios simulados para la medida de la desviación media absoluta.

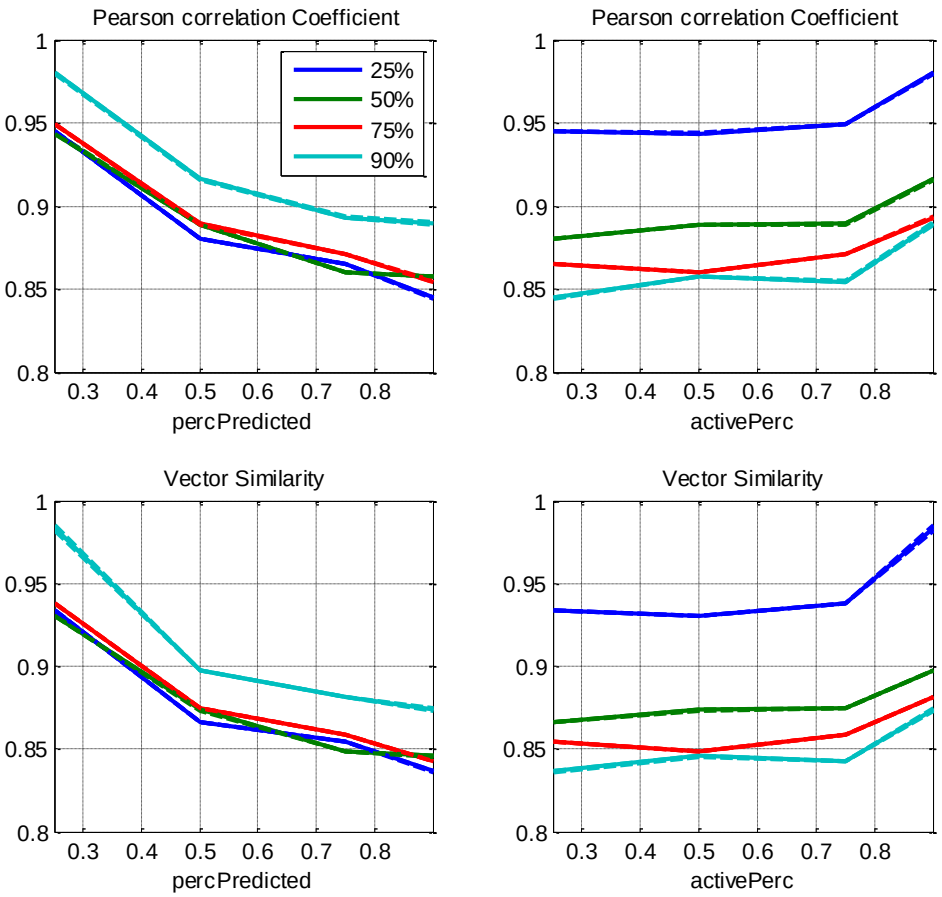

Figura 4.9 Mean absolute deviation en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo basados en memoria usando el coeficiente de correlación de Pearson y Vector similarity

Los sistemas de recomendación mediante filtrado colaborativo basados en memoria dan sus mejores resultados cuando el número de usuarios activos en el sistema está entre un porcentaje inferior al 75% del total. Cuando el número de usuarios activos crece hasta el 90% las prestaciones empiezan a empeorar. Del mismo modo presentan mejor rendimiento cuando los usuarios activos han visto una cantidad elevada de películas, y empeoran sus resultados cuando todavía han visto pocas películas. También en el caso contrario, cuando ha visto muchas películas, por lo que es difícil encontrar contenido no visionado del agrado del consumidor a recomendar.

También tenemos que destacar que aunque ambos sistemas tienen una tendencia similar, los sistemas de recomendación mediante filtrado colaborativo basados en memoria que usan coeficiente de correlación de Pearson son peores que los que usan Vector Similarity, y experimentalmente hemos podido comprobar que son computacionalmente similares.

Pasamos ahora a comentar los resultados obtenidos por los sistemas de recomendación mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias entre usuarios.

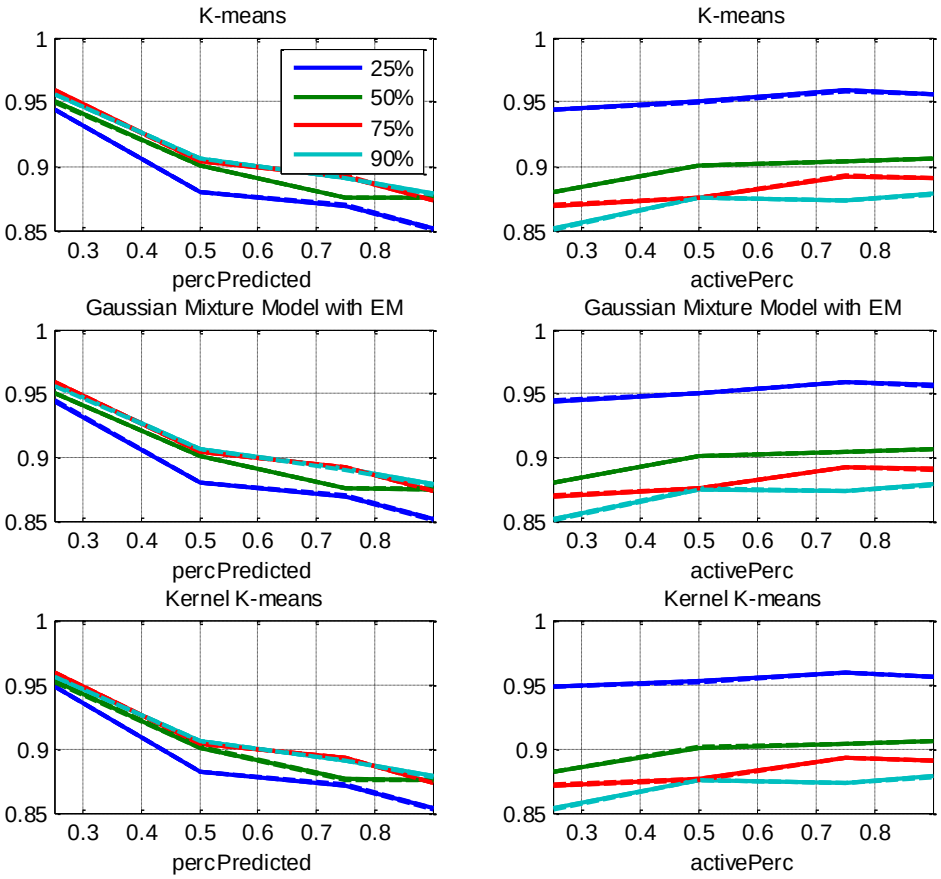

Figura 4.10 Mean absolute deviation en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo utilizando un 5% del total de usuarios como centroides y los métodos de clustering

Como podemos ver los algoritmos que usan la distancia euclídea a la hora de calcular la distancia de cada dato a los centroides proporcionan resultados similares. Esto puede ser debido a que aunque usemos tres técnicas de clustering diferentes, el subespacio de centroides por películas que se crea es equivalente, por lo que cuando calculamos la matriz de similitud a partir de la distancia de los usuarios a los centroides estas sean iguales. Es por eso que sería interesante como línea de trabajo futura cambiar de modo para calcular la distancia de cada dato a los centroides, aplicando por ejemplo, el algoritmo K-medias ponderado usando como referente la distancia coseno, obteniendo así un subespacio diferente y por lo tanto posibles resultados diferentes.

4.4.2 Mean Square Error

Ahora pasamos a comentar el error cuadrático medio. La salida de los sistemas de recomendación de esta medida de evaluación es similar a la de MAD. Nos interesa un MSE lo más bajo posible, ya que así la diferencia entre el valor estimado y el real es menor.

Como antes, los sistemas de recomendación mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias entre usuarios empeoran los sistemas de recomendación mediante filtrado colaborativo basados en memoria por lo general.

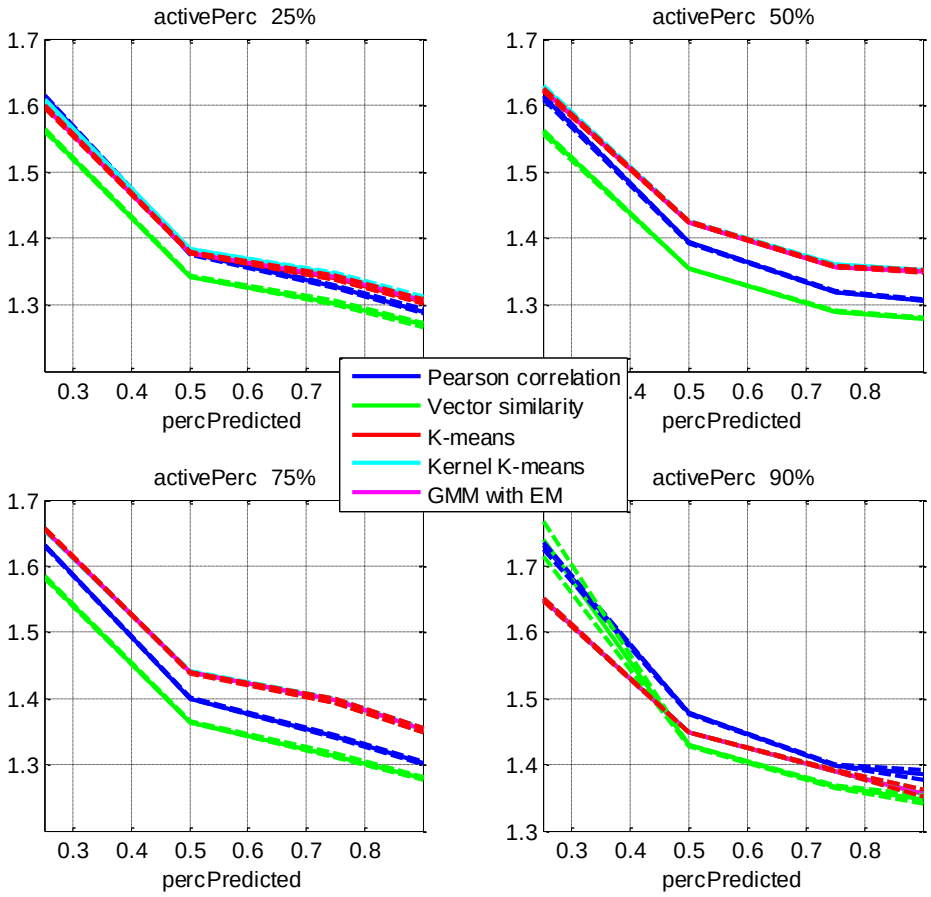

Figura 4.11 Medida mean square error manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 5% del total de usuarios como centroides

Observando la figura 4.11, podemos decir que desde el punto de vista del MSE sería interesante usar sistemas de recomendación mediante filtrado colaborativo basados en memoria cuando todavía no tenemos muchas votaciones de usuarios y hay muchos usuarios activos para recomendar nuevo contenido a usuarios que ya han visto una cantidad alta de películas en el sistema. Esto se puede ver exactamente para un 90% de usuarios activos para los que los algoritmos de agrupamiento en un porcentaje de películas vistas/predichas por los usuarios activos del 25% mejoran al sistema de recomendación que usa Vector Similarity y mejoran o igualan en todo caso al coeficiente de correlación de Pearson.

Comparando métodos de clustering entre sí vemos que todos dan prestaciones muy similares.

Al igual que cuando presentamos los resultados de la MAD también hemos creado una figura donde se vea mejor la evolución de los sistemas de recomendación en función del incremento de usuarios activos en función de su cantidad de películas vistas.

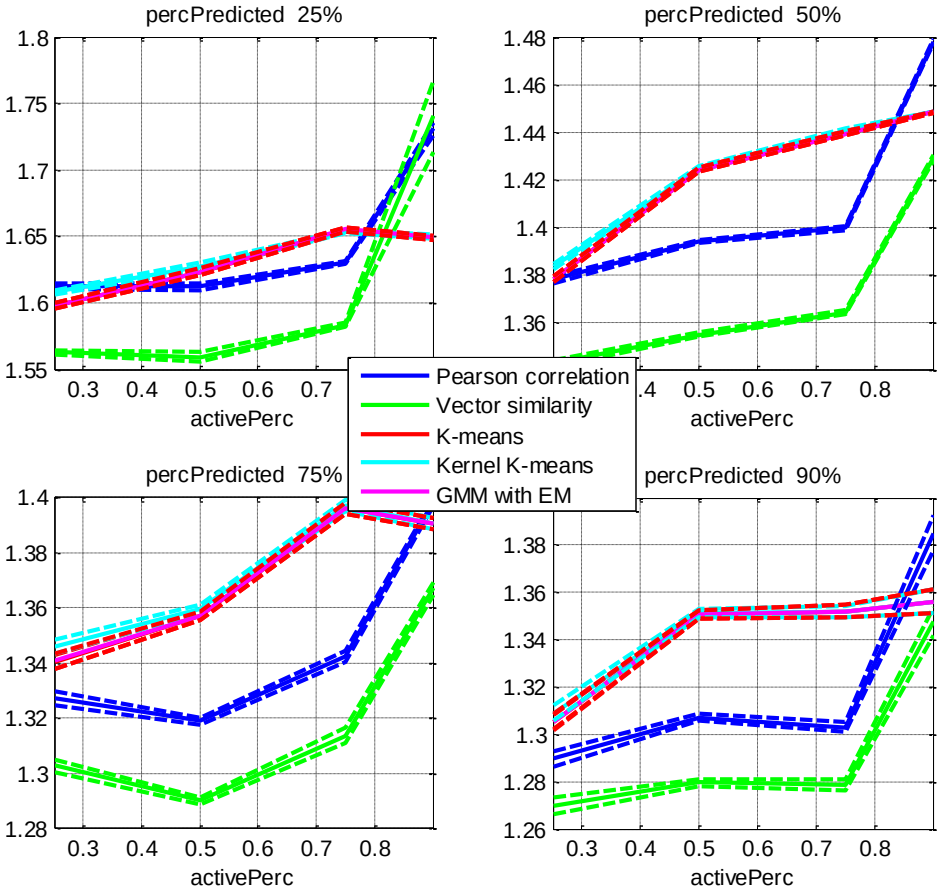

Figura 4.12 Medida mean square error manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 5% del total de usuarios como centroides

Desde el punto de vista del MSE no podemos decir nada nuevo de lo ya comentado acerca de la MAD. Los métodos de agrupamiento tienen peores prestaciones que los sistemas de recomendación clásicos, pero al ver la curva de sus graficas nos damos cuenta que sus resultados son más estables, ya que vemos que las pendientes de sus curvas son más suaves.

El resto de simulaciones que presenten los mismos resultados irán una a continuación de otra y serán comentadas conjuntamente para mayor comodidad a la hora de interpretar los resultados matemáticos obtenidos.

En las figuras 4.13 y 4.14 pueden verse ahora el comportamiento habiendo incrementado al 10% del total de usuarios el número de centroides.

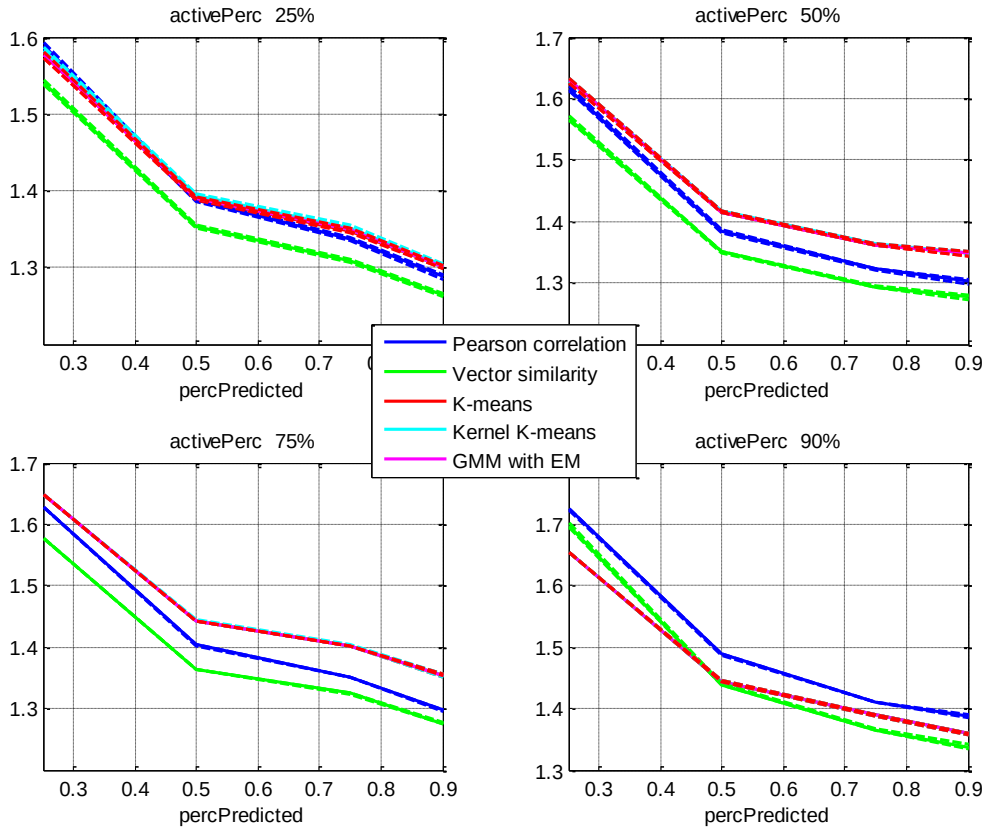

Figura 4.13 Medida mean square error manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 10% del total de usuarios como centroides

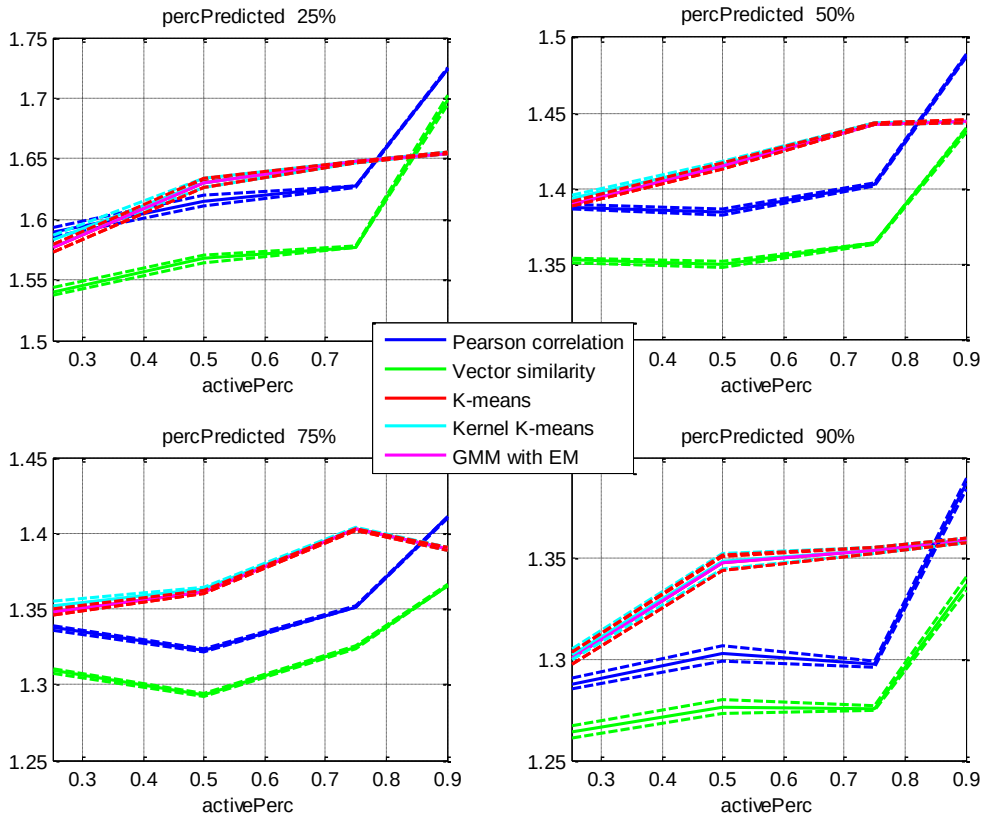

Figura 4.14 Medida mean square error manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 10% del total de usuarios como centroides

Duplicando los centroides con respecto al escenario previo los sistemas de recomendación mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias entre usuarios dan resultados similares entre ellos, y más próximos a los basados en memoria.

Para un porcentaje del 25% de usuarios activos, los métodos de agrupamiento son mejores que usar el coeficiente de correlación de Pearson cuando el usuario ha visto pocas películas todavía, y son un poco peores que vector similarity para un porcentaje del 90% de usuarios activos y pocas películas vistas. Estos resultados son interesantes para intentar de solucionar problemas asociados a los sistemas de recomendación como el Cold-Start.

En las figuras 4.15 y 4.16 pueden verse ahora el comportamiento habiendo fijado el número de centroides al 20% del total de usuarios del sistema.

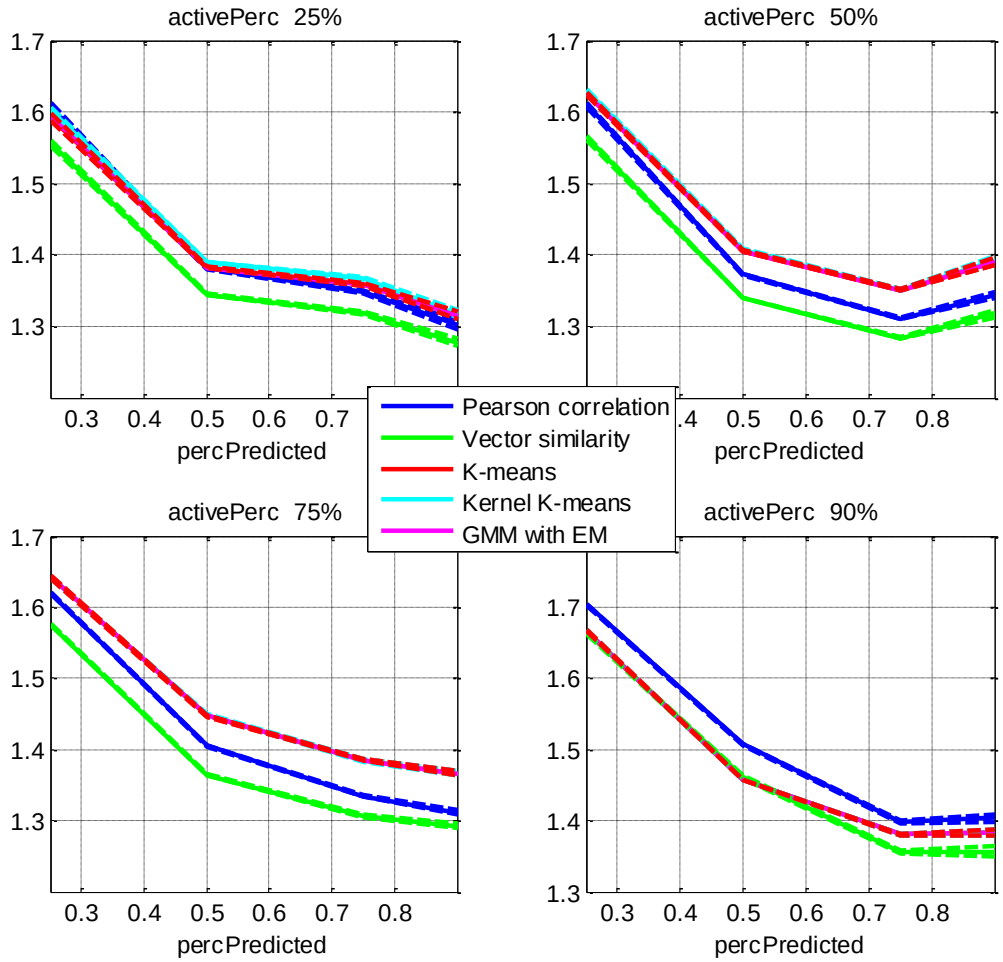

Figura 4.15 Medida mean square error manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 20% del total de usuarios como centroides

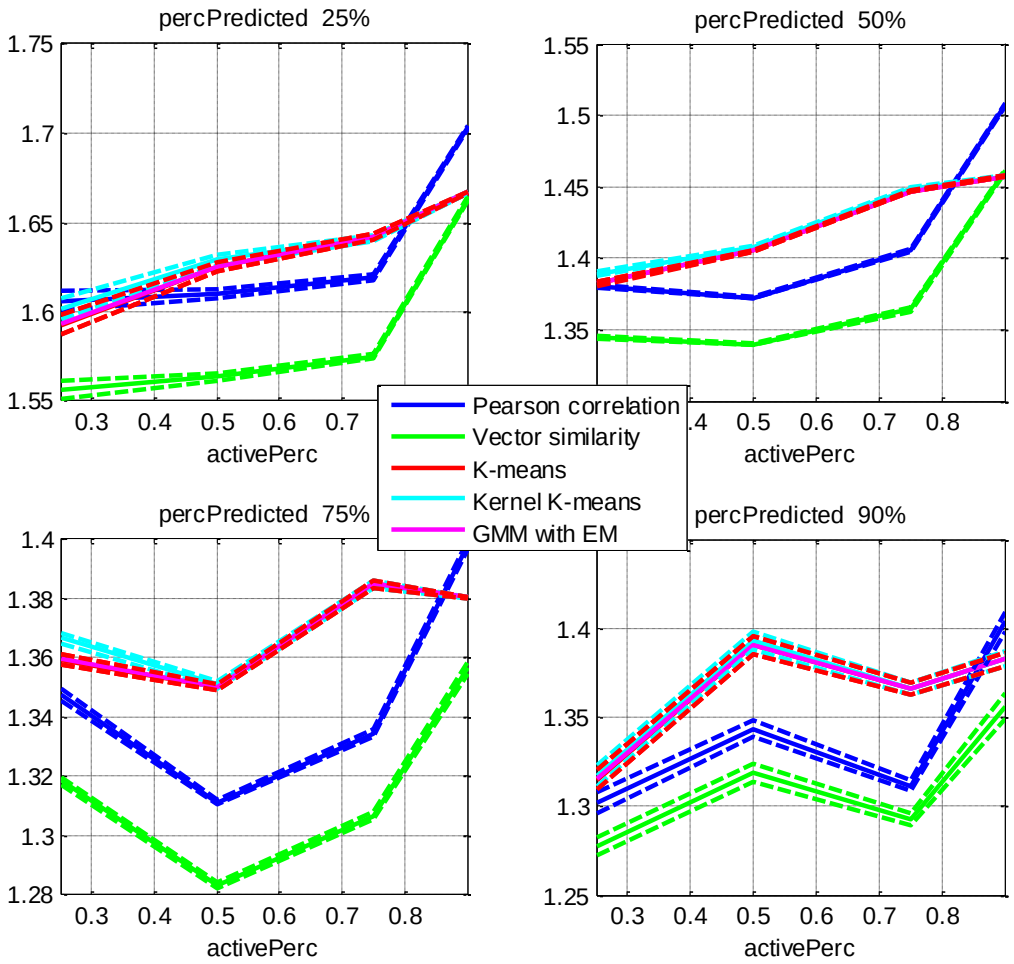

Figura 4.16 Medida mean square error manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 20% del total de usuarios como centroides

Al aumentar a 20% del total de usuarios el número de centroides para un porcentaje del 25% de usuarios activos, los métodos de agrupamiento alcanzan en prestaciones a vector similarity para un porcentaje del 90% de usuarios activos y pocas películas vistas.

Comparando métodos de clustering entre sí los resultados se igualan mucho, continuando siendo el peor Kernel k-means, por lo que nuestras suposiciones acerca de que éste es un porcentaje adecuado de centroides se reafirman.

En las figuras 4.17 y 4.18 pueden verse ahora el comportamiento habiendo fijado el número de centroides al máximo simulado, es decir, al 30% del total de usuarios del sistema. Con el máximo de centroides simulados la situación no varía demasiado, por lo que lo ya expuesto para el caso anterior es aplicable a este.

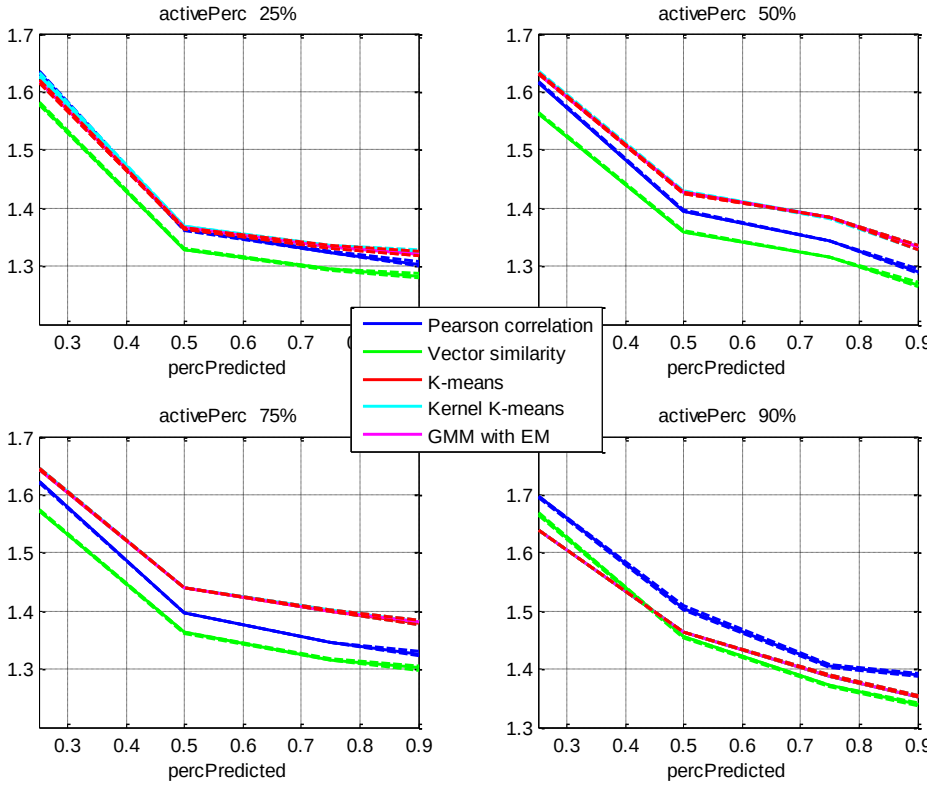

Figura 4.17 Medida mean square error manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 30% del total de usuarios como centroides

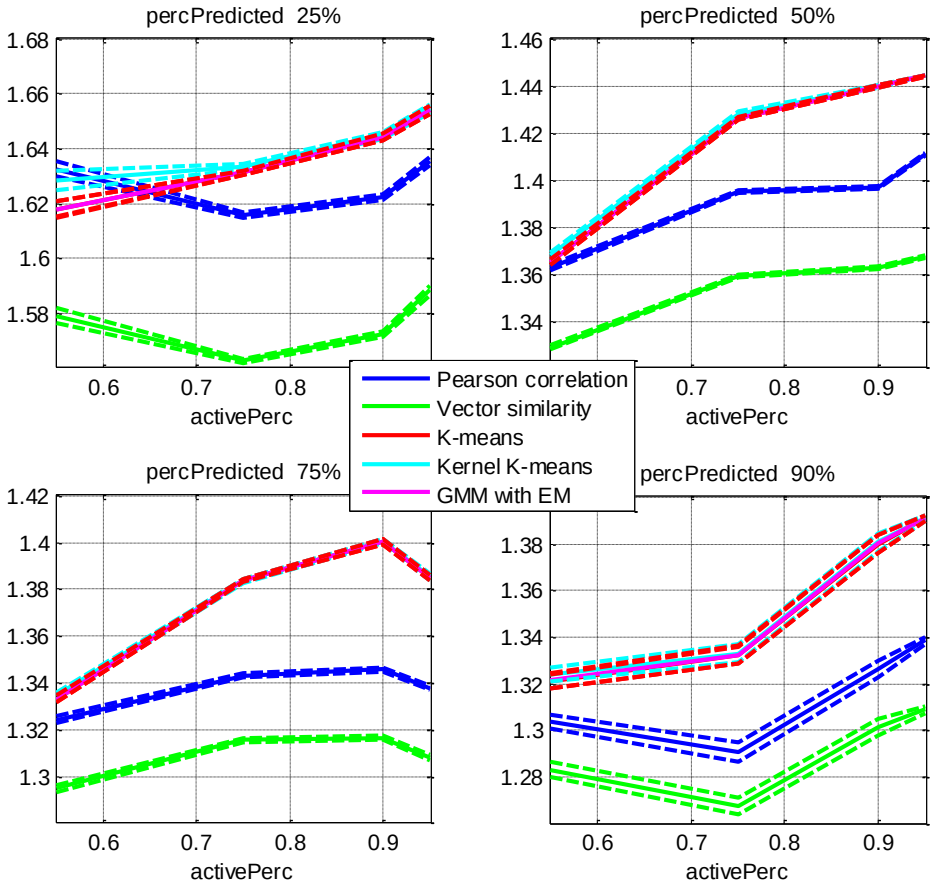

Figura 4.18 Medida mean square error manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 30% del total de usuarios como centroides

Ahora se va a pasar a evaluar individualmente cada método en los escenarios simulados para la medida del error cuadrático medio.

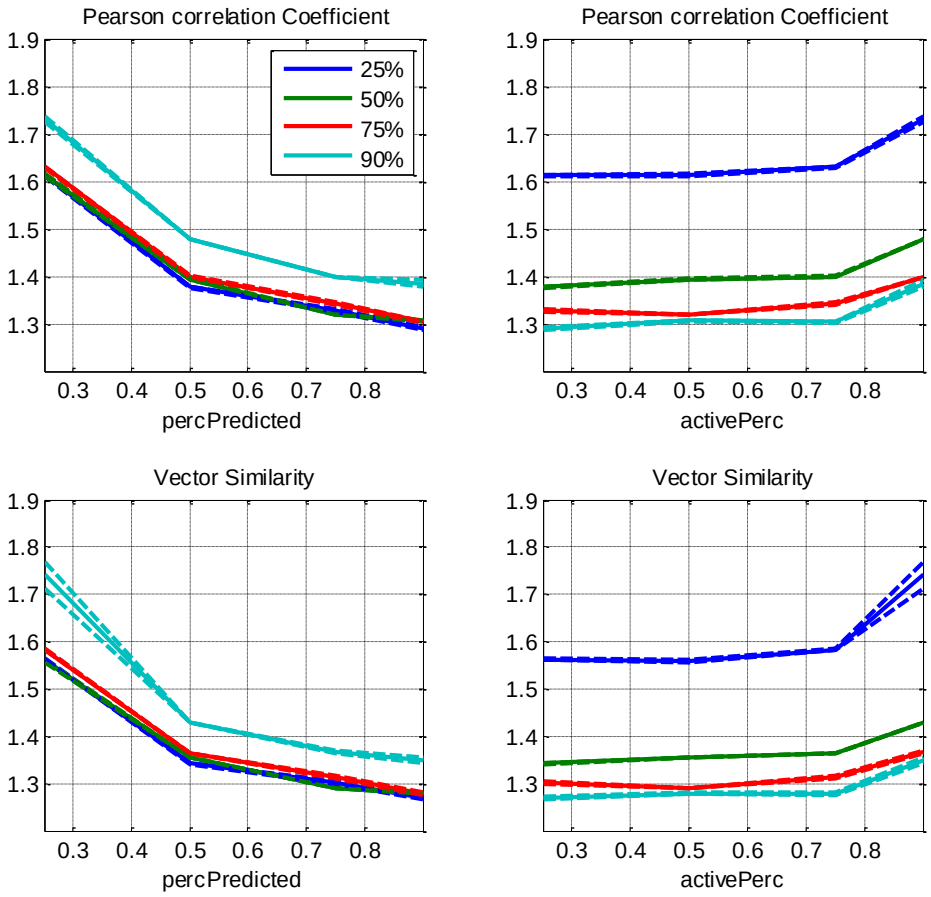

Figura 4.19 Medida mean square error en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo basados en memoria usando el coeficiente de correlación de Pearson y Vector similarity

Desde el punto de vista de la MSE al igual que pasaba con la MAD, los sistemas de recomendación mediante filtrado colaborativo basados en memoria obtienen mejores resultados cuando el número de usuarios activos en el sistema está entre un 25% y un 75%. A diferencia de la MAD, la MSE no ve incrementado su error por reducir del 75% al 50% el número de usuarios de entrenamiento, por lo que si quisiéramos disminuir la carga computacional podríamos usar esta medida de evaluación sin perder prestaciones.

Los sistemas de recomendación mediante filtrado colaborativo basados en memoria que usan coeficiente de correlación de Pearson son peores que los que usan Vector Similarity y computacionalmente igual de costosos, aunque la varianza de utilizar Vector Similarity es mayor.

Pasamos ahora a comentar los resultados obtenidos por los sistemas de recomendación mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias entre usuarios.

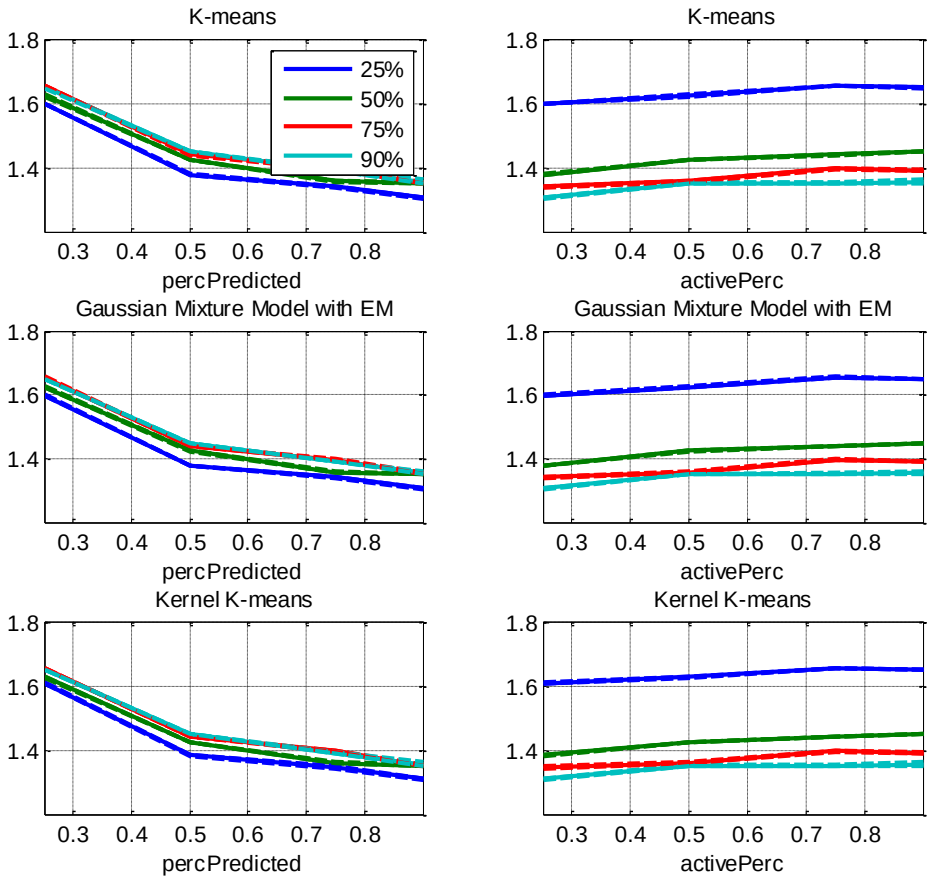

Figura 4.20 Medida mean square error en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo utilizando un 30% del total de usuarios como centroides y los métodos de clustering

La discusión hecha para la MAD también es aplicable a la MSE. Los algoritmos que usan la distancia euclídea a la hora de calcular la distancia de cada dato a los centroides proporcionan resultados similares posiblemente debido a que crean subespacios de centroides por películas equivalentes y por consiguiente matrices de similitud parecidas.

4.4.3 Ranked Evaluation

Aunque el estudio del error cuadrático medio y de la desviación media son interesantes porque son parámetros de evaluación muy usados, sencillos y estudiados, es necesario introducir una medida de evaluación propia de los sistemas de recomendación. Es por eso que empleamos el ranked evaluation para medir preferencias reales de los usuarios.

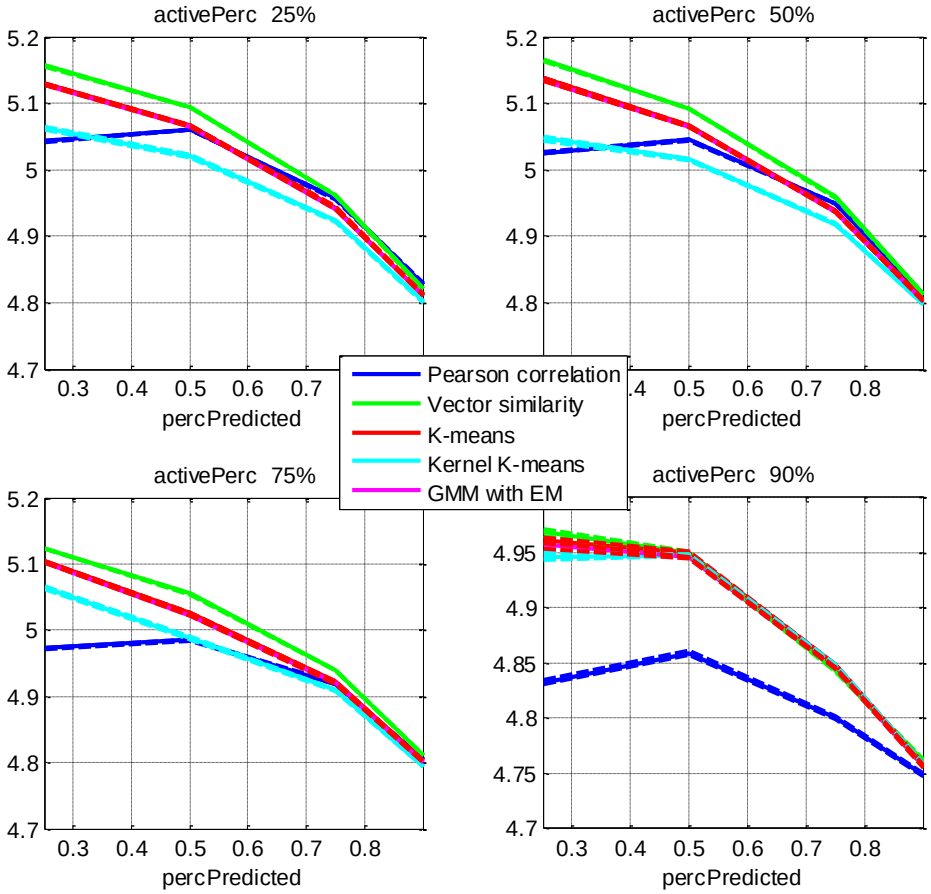

Figura 4.21 Medida ranked evaluation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 5% del total de usuarios como centroides

En un primer lugar vemos que todas las curvas siguen la misma tendencia. Ésta es lógica porque cuando el usuario ha visionado un número bajo pero suficiente de películas la lista de posibles recomendaciones es mayor y, por tanto, más películas que a él le gustaría ver por ser de un tema relacionado caen dentro de esa lista de recomendaciones.

Al principio vemos que cuando todavía no se han visto un número suficiente de películas se produce el problema de Cold-Start. Este afecta en mayor medida a los sistemas de recomendación que usan el coeficiente de correlación de Pearson y Kernel K-medias. Por el contrario los métodos de clustering con K-medias ponderado obtienen resultados similares a los de Vector Similarity.

A medida que aumentan el número de contenido visto la mayoría de soluciones, salvo el coeficiente de correlación de Pearson, convergen a los mismos resultados.

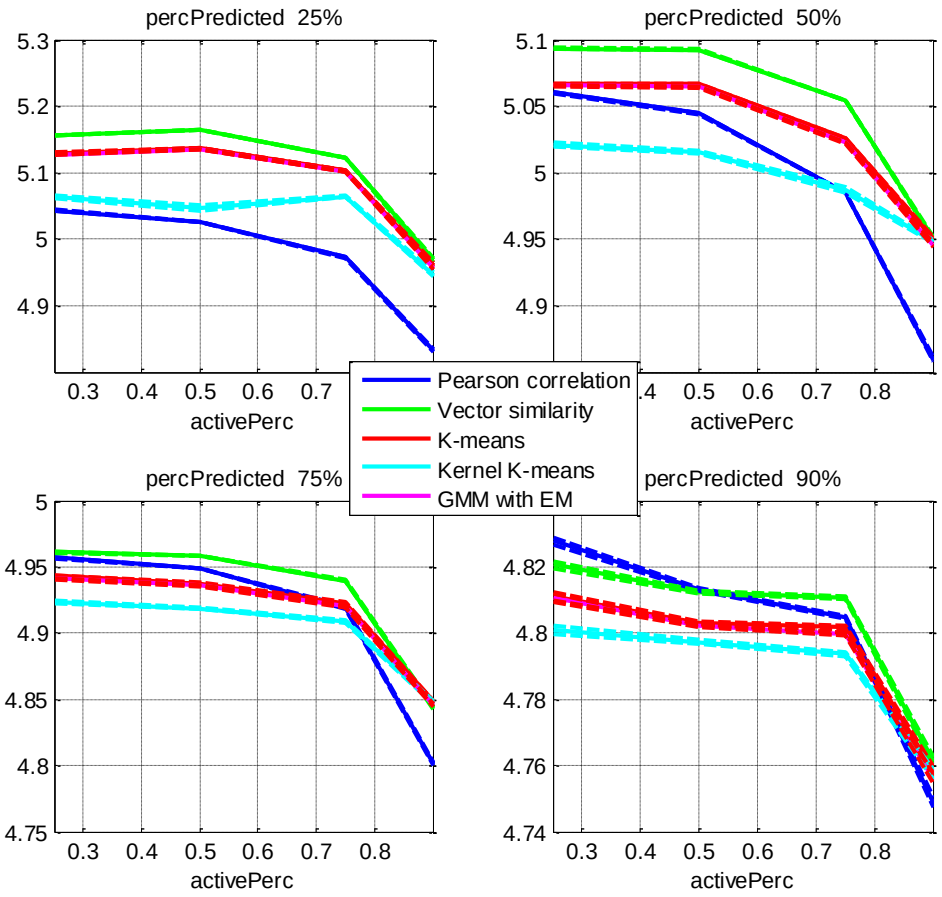

Figura 4.22 Medida ranked evaluation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 5% del total de usuarios como centroides

En la figura 4.22 es interesante ver el número de usuarios que necesitaríamos con votaciones a la hora de evaluar al usuario activo. Cuando hemos simulado usuarios con un 75% de películas ya vistas (conjunto de test) y el 25% restante para evaluar prestaciones (conjunto de validación) el rendimiento de todos los sistemas de recomendación es parejo. Para un menor número de películas vistas algunos sistemas de recomendación necesitan una mayor base de datos que otros para converger al mismo resultado o incluso no lo consiguen como en el caso de los SR que emplean el coeficiente de correlación de Pearson.

El resto de simulaciones que presenten los mismos resultados irán una a continuación de otra y serán comentadas conjuntamente para mayor comodidad a la hora de interpretar los resultados matemáticos obtenidos.

En las figuras 4.23 y 4.24 pueden verse ahora el comportamiento habiendo doblado el número de centroides.

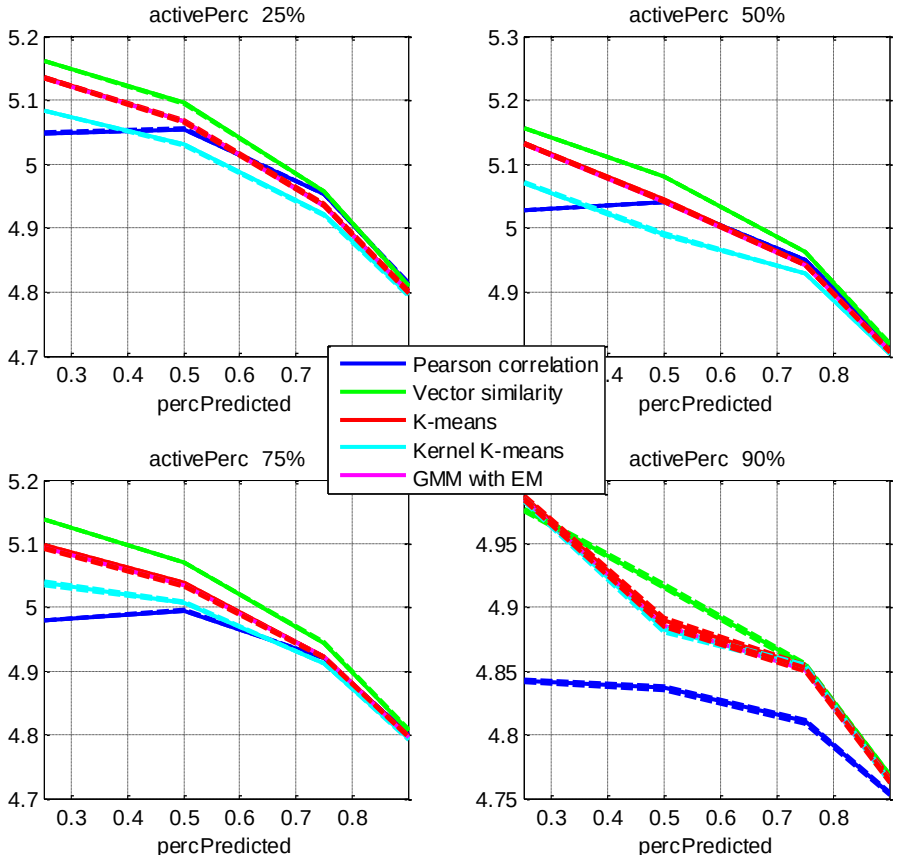

Figura 4.23 Medida ranked evaluation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 10% del total de usuarios como centroides

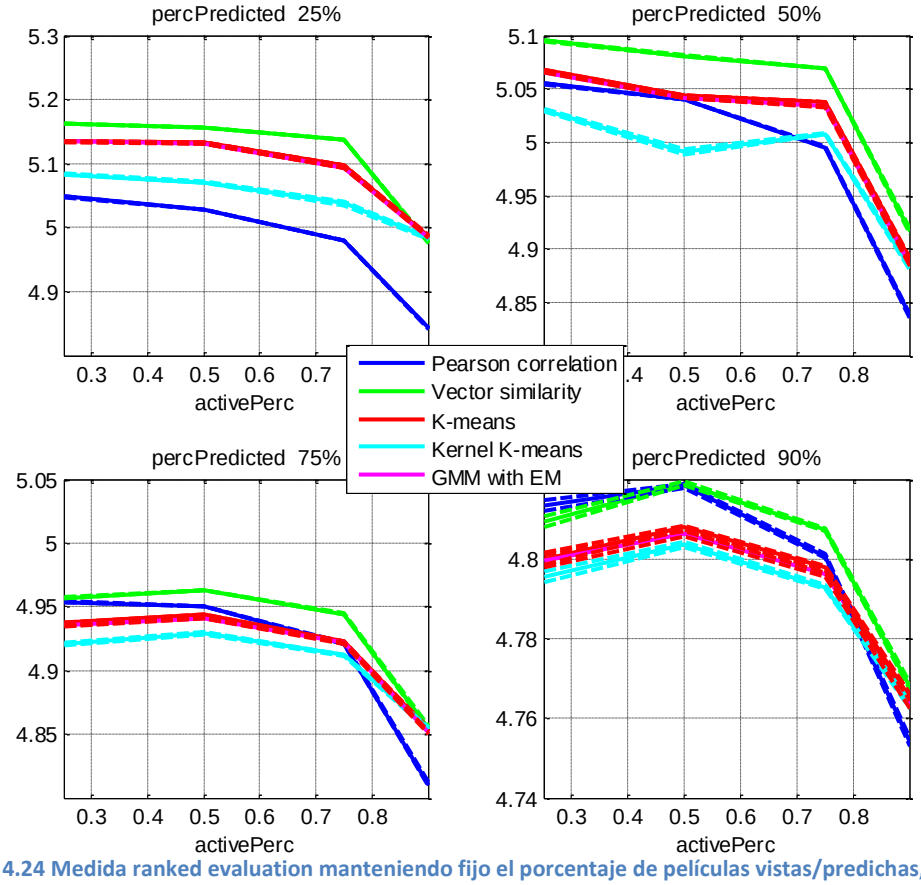

Figura 4.24 Medida ranked evaluation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 10% del total de usuarios como centroides

A partir de ahora nos centraremos en comentar la repercusión que implica aumentar el número de centroides en la salida de nuestros SR basados en métodos de agrupamiento.

La duplicación del número de centroides hace que los resultados obtenidos por los métodos de agrupamiento a partir de un porcentaje de películas vistas/predichas del 75% sean prácticamente iguales a los de usar vector similarity. Frente a usar el coeficiente de correlación de Pearson se mejora en la gran mayoría de los escenarios.

El método clustering Kernel K-medias obtiene peores prestaciones que los otros dos algoritmos de agrupamiento.

En las figuras 4.25 y 4.26 pueden verse ahora el comportamiento habiendo fijado el número de centroides al 20% del total de usuarios del sistema.

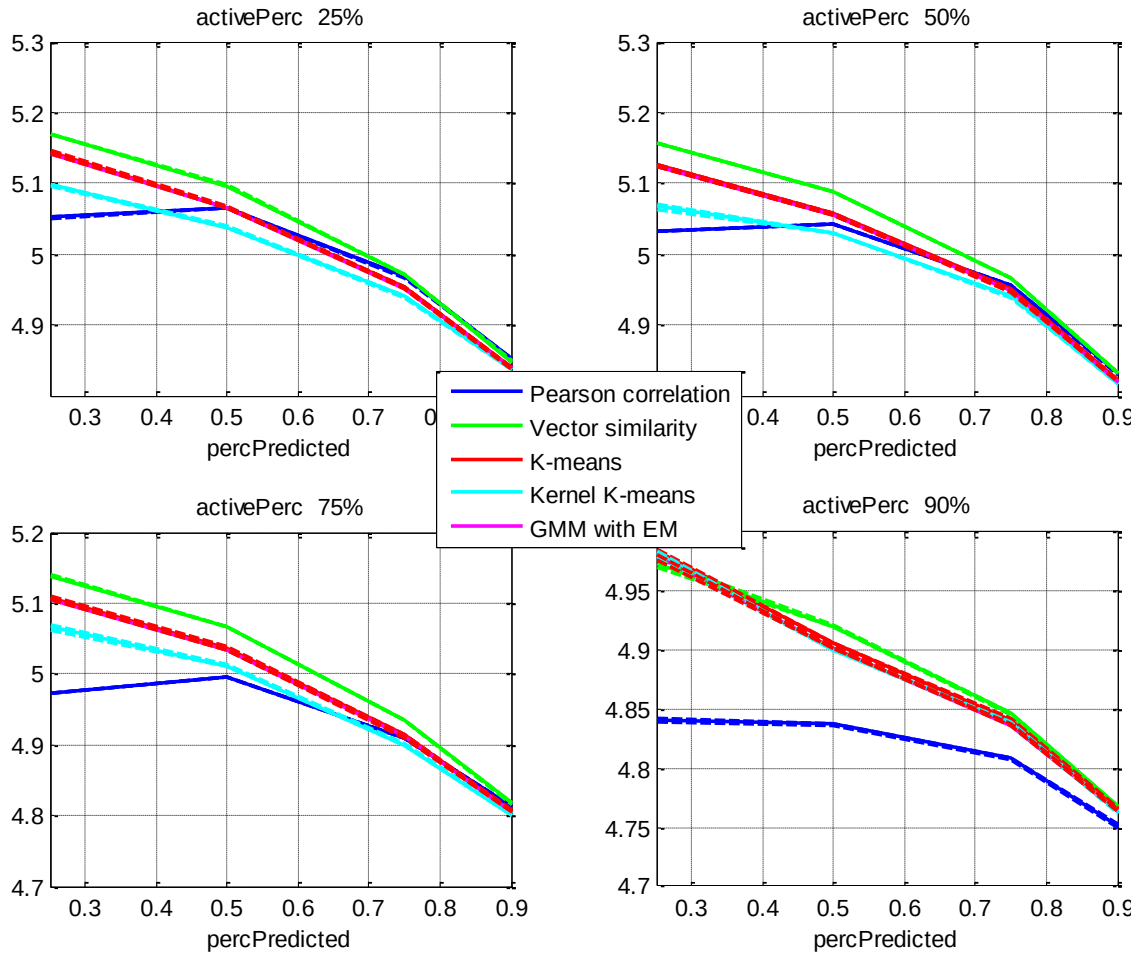

Figura 4.25 Medida ranked evaluation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 20% del total de usuarios como centroides

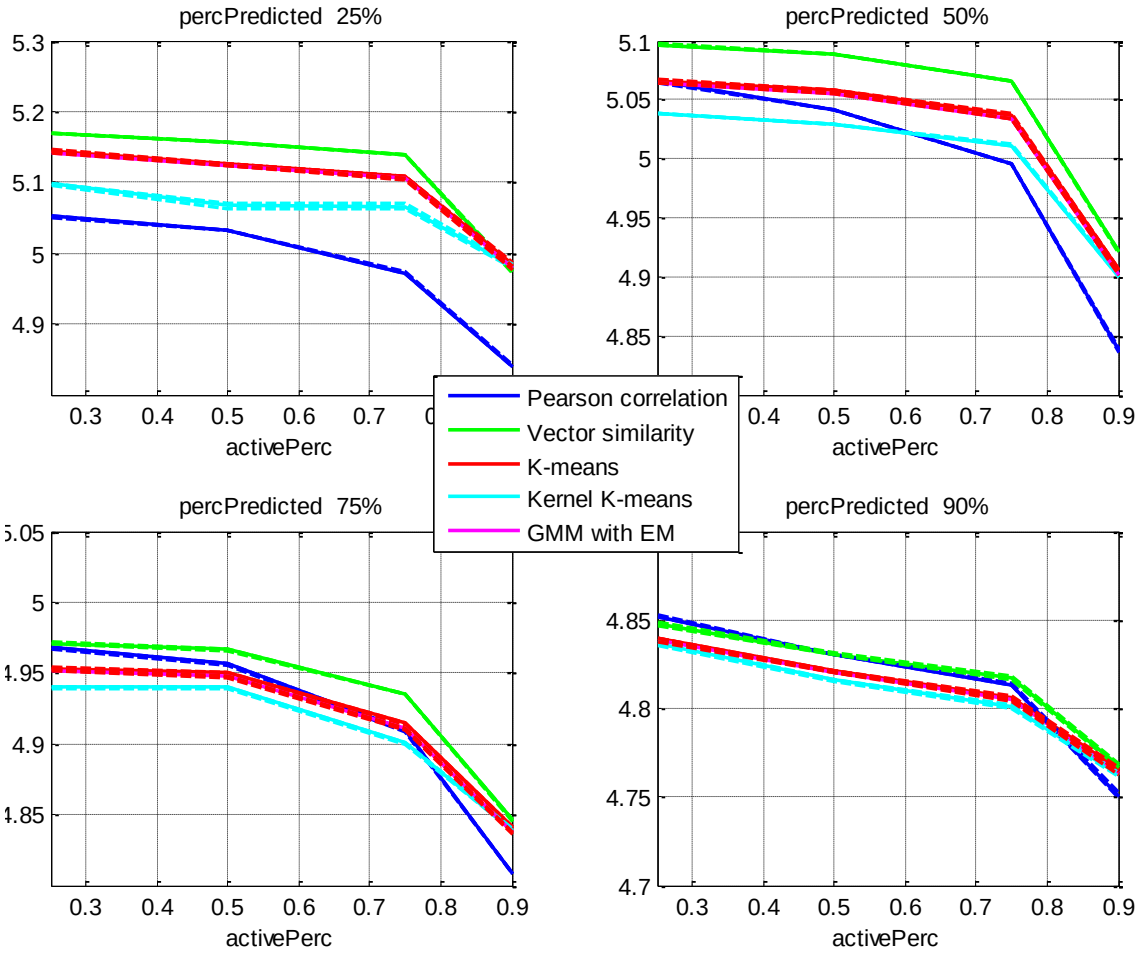

Figura 4.26 Medida ranked evaluation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 20% del total de usuarios como centroides

Un nuevo incremento del número de centroides hasta el 20% del total de usuarios no provoca grandes cambios en los algoritmos de agrupamiento, por lo que para esta medida de evaluación con usar un 10% del total de usuarios en el sistema como número de centroides sería suficiente.

En las figuras 4.27 y 4.28 pueden verse ahora el comportamiento habiendo fijado el número de centroides al máximo simulado, es decir, al 30% del total de usuarios del sistema. Con el máximo de centroides simulados la situación no varía demasiado, por lo que lo ya expuesto para el caso anterior es aplicable a este.

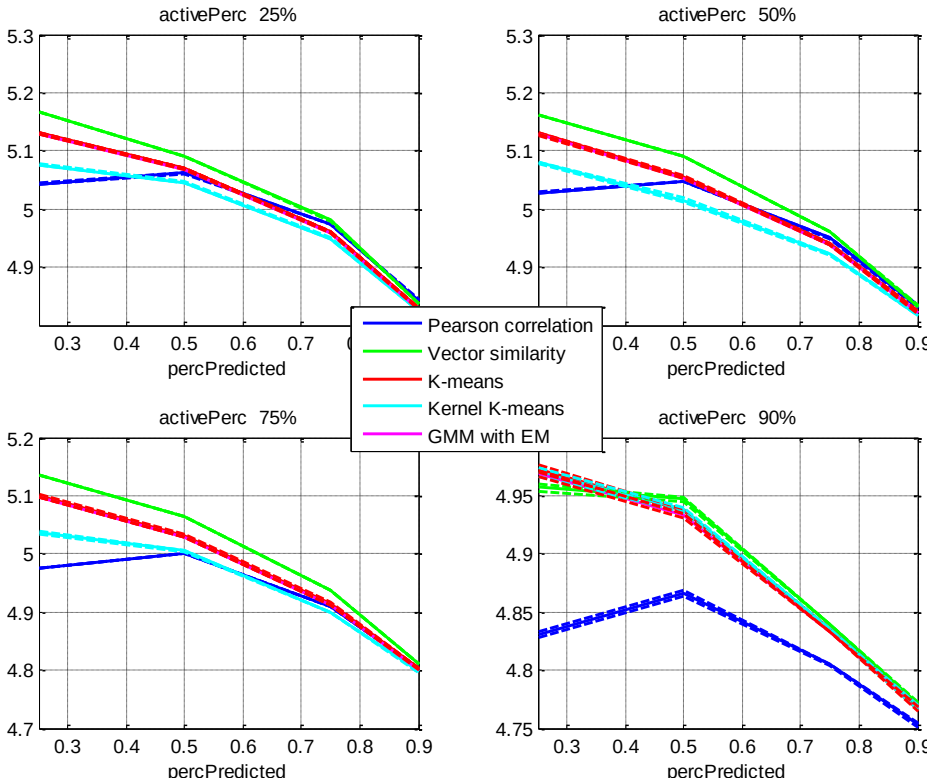

Figura 4.27 Medida ranked evaluation manteniendo fijo el número de usuarios activos, variando el porcentaje de películas vistas/predichas por los usuarios activos y usando un 30% del total de usuarios como centroides

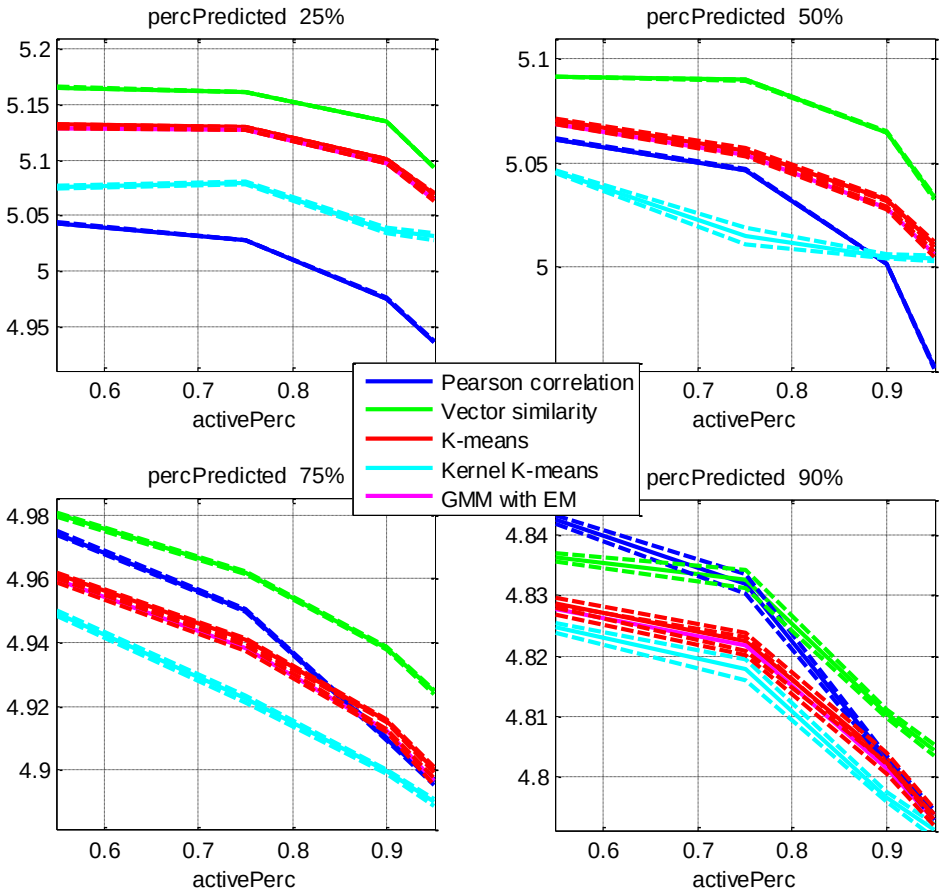

Figura 4.28 Medida ranked evaluation manteniendo fijo el porcentaje de películas vistas/predichas, variando la proporción de usuarios activos y usando un 30% del total de usuarios como centroides

Ahora se va a pasar a evaluar individualmente cada método en los escenarios simulados para la medida ranked evaluation.

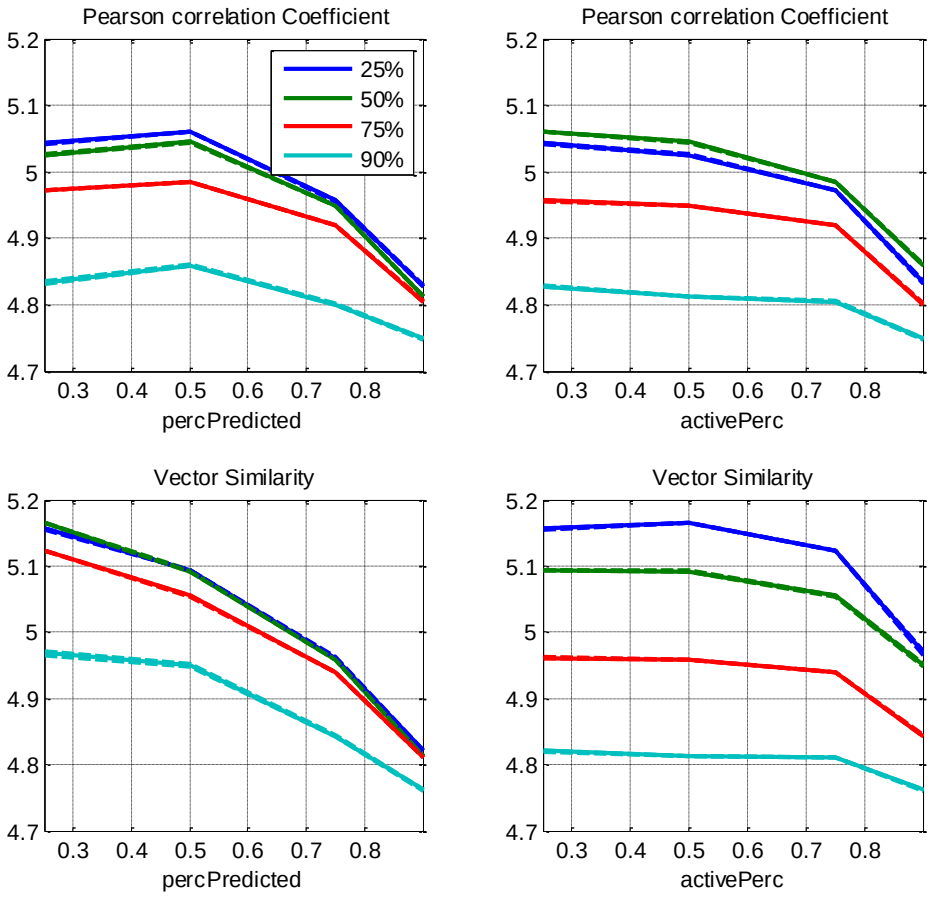

Figura 4.29 Medida ranked evaluation en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo basados en memoria usando el coeficiente de correlación de Pearson y Vector similarity

Comparando los resultados obtenidos por Ranked Evaluation para los SR mediante filtrado colaborativo basados en memoria vemos que empleando coeficiente de correlación de Pearson obtenemos en general peores recomendaciones. La mayor diferencia se produce para usuarios con pocas películas vistas independientemente del tamaño de la base de datos existente. Por ello no es recomendable usar este SR a la hora de hacer recomendaciones a usuarios nuevos ya se sufre de Cold-Start.

Pasamos ahora a comentar los resultados obtenidos por los sistemas de recomendación mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias entre usuarios.

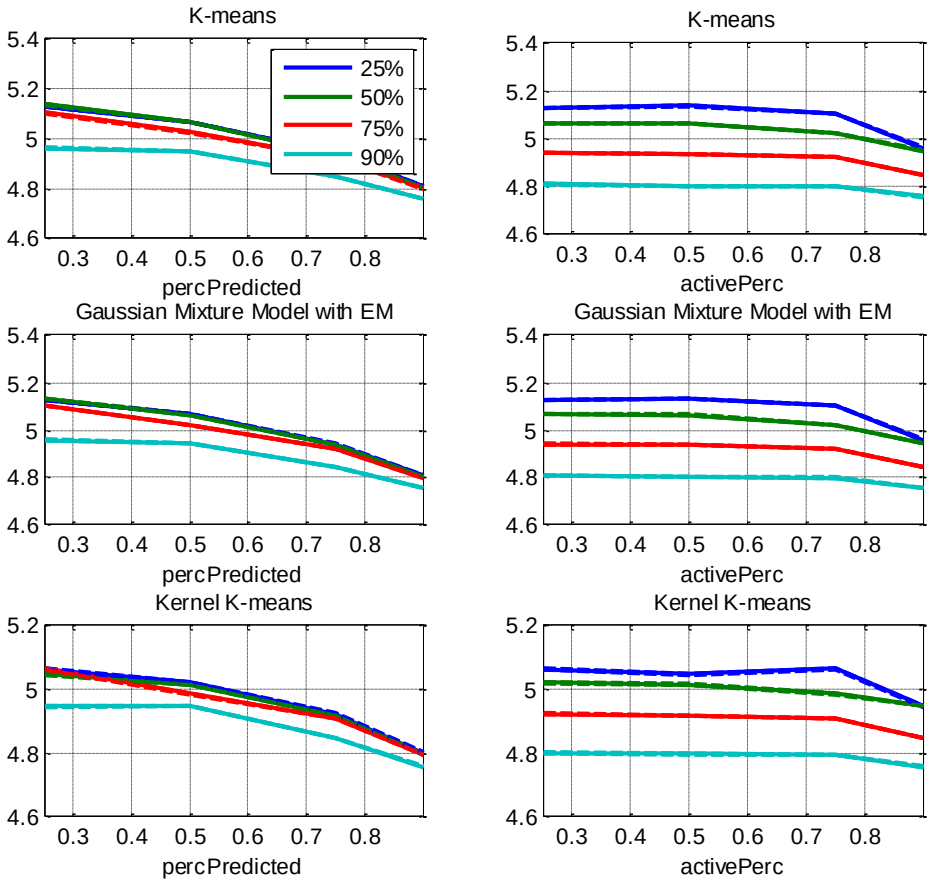

Figura 4.30 Medida ranked evaluation en todos los escenarios simulados para los sistemas de recomendación mediante filtrado colaborativo utilizando un 30% del total de usuarios como centroides y los métodos de clustering

Los SR mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias entre usuarios obtienen en general buenas prestaciones si los comparamos con los otros dos sistemas de recomendación presentados.

El método de mezcla de gaussianas entrenadas con EM es el que peor se adapta a la base de datos empleada. El resto de los algoritmos de clustering, como ya hemos comentado en graficas anteriores, mejoran la salida del recomendador que usa coeficiente de correlación de Pearson e iguala las prestaciones de Vector Similarity. El uso de K-medias ponderado se ha posicionado como una interesante alternativa a la hora de calcular la distancia entre usuarios dentro de los sistemas de recomendación gracias a su simplicidad y rendimiento.

5 CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

5.1 Conclusiones

En este proyecto fin de carrera nos hemos centrado en la inserción de métodos de agrupamiento en el cálculo de distancias entre usuarios en un sistema de recomendación basado en algoritmos de filtrado colaborativo. Hemos podido evaluar la repercusión en el comportamiento y las prestaciones de los algoritmos de agrupamiento presentados en el apartado 3 en los sistemas de recomendación.

Los métodos de clustering que experimentalmente hemos probado han sido el algoritmo de K-medias ponderado usando distancia euclidea, Kernel K-medias ponderado usando un kernel del tipo polinómico y Mezcla de Gaussianas entrenadas con EM.

En el capítulo anterior hemos presentado y discutido los resultados experimentales en varios escenarios trabajando sobre la base de datos de EachMovie. Como hemos podido comprobar, los resultados no han sido tan buenos como esperábamos a priori, pero todo el trabajo experimental desarrollado es un buen punto de partida para continuar explorando este campo de investigación.

La utilidad de todo el trabajo realizado aplicado a sistemas reales cabe ser destacado. La inserción de métodos de agrupamiento sobre el conjunto de usuarios crea *Prototipos de Usuarios*, los cuales unifican las preferencias de un conjunto de usuarios en esos usuarios tipo. Por ejemplo, desde el punto de vista de recomendación de películas se podrían generar usuarios prototipo por cada uno de los géneros cinematográficos existentes: cine de acción, ciencia ficción, cine negro, comedia romántica, cine de terror, cine clásico, cine musical, cine de aventuras… Y usar los gustos de estos usuarios prototipo para conseguir recomendaciones generales sobre un género concreto.

Por lo tanto, con estas ideas presentadas se podrían buscar nuevas alternativas para solucionar algunos de los problemas que sufren los sistemas de recomendación basados en algoritmos de filtrado colaborativo, como son el Cold-Start o el Early Adopter.

5.2 Líneas Futuras de Trabajo

En esta última sección se van a presentar posibles líneas de trabajo para continuar todo lo presentado en este proyecto fin de carrera:

- Inserción de nuevos algoritmos de agrupamiento para el cálculo de distancias entre usuarios, como por ejemplo, emplear distribuciones multinomiales, Fuzzy c-means clustering o Quality Threshold clustering.

- Estudiar el comportamiento y las prestaciones de los métodos presentados para usuarios particulares en lugar de escenarios generales, comprobando por ejemplo, cual es el acierto condicionado al número de votaciones para ver que algoritmos se comportan mejor cuando tenemos problemas de Cold-Start, Sparsity…

- Un trabajo interesante que se plantea en los sistemas de recomendación es combinar la técnica de filtrado colaborativo con la basada en contenido de forma de obtener los beneficios de cada una. En este sentido se podría agregar análisis de contenido de los elementos del sistema de forma de poder mejorar las recomendaciones así como también asistir en la etapa de Cold Start del usuario.

- Empleo de los data sets proporcionados en el apartado 2.1.6.2 para evaluar los sistemas de recomendación mediante filtrado colaborativo usando métodos de agrupamiento en el cálculo de distancias en diferentes bases de datos: libros, música y películas…

6 PRESUPUESTO

En el presupuesto que se presenta a continuación se hace un estudio estimado de los costes de realización de este proyecto. Se han considerado los costes relativos al material, licencias software y personal.

6.1 Costes de personal

Los costes que se van a detallar a continuación corresponden a aquellos desprendidos en las etapas de desarrollo, medida y redacción de la documentación relativa al proyecto.

Haciendo una estimación de la duración útil de este proyecto se podría decir que ha sido de 3 meses invirtiendo 5 horas al día en su realización, exceptuando fiestas y vacaciones. Por los tanto, los costes de personal ascienden a:

Trabajador	Horas	Coste Hora	Coste Total
Ingeniero de Telecomunicaciones	450 horas	15€/hora	6750€

Figura 6.1 Presupuesto para los costes de personal

A partir de los datos anteriores se establece que los gastos personales disgregados, describiendo el concepto y número de horas de trabajo, son:

Concepto	Horas	Coste Hora	Coste Total
Investigación y formación	50 horas	15€/hora	750€
Trabajo experimental	200 horas	15€/hora	3000€
Preparación y redacción del documento	200 horas	15€/hora	3000€

Figura 6.2 Desglose del presupuesto de los costes de personal

6.2 Costes de material

En los costes relativos al material se van a incluir los costes de las licencias del software utilizado y los gastos asociados a las medidas realizadas, incluyendo los costes de los equipos. No se incluyen los costes de amortización de los equipos.

Concepto	Cantidad	Coste Unidad	Coste Total
Licencia Software Matlab	1	500€/unidad	500€
Licencia Software Microsoft Office	1	139€/unidad	139€
PC de Sobremesa	1	600€/unidad	600€
Total			**1239€**

Figura 6.3 Presupuesto para los costes de material

El precio de la licencia de Matlab corresponde a una licencia para uso académico y ha sido obtenido de la página web: *www.mathworks.es*

El precio de la licencia de Microsoft Office Hogar y Estudiantes 2010 corresponde a una licencia para uso académico y ha sido obtenido de la página web: http://office.microsoft.com

* En el caso de que fuera necesario la impresión y encuadernación de la memoria habría que añadir un coste extra de 300€, a razón de 100€ por cada una de las copias a entregar.

6.3 Presupuesto total

Una vez calculados los costes relativos al personal y al material, el presupuesto total de este proyecto fin de carrera asciende a **siete mil novecientos ochenta y nueve** euros:

Costes	Presupuesto
Costes de material	1239€
Costes de personal	6750€
TOTAL	**7989€**

Figura 6.4 Presupuesto para los costes totales

7 BIBLIOGRAFÍA

[1] Guy Lebanon. (2003) C/Matlab Toolkit for Collaborative Filtering. [Online]. http://www.cc.gatech.edu/~lebanon/software/CFtoolbox/

[2] Lev Grossman. (2010) How Computers Know What We Want — Before We Do. [Online]. http://www.time.com/time/magazine/article/0,9171,1992403-1,00.html

[3] I. S. Dhillon, Y. Guan, and B. Kulis, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 29:11, pages 1944-1957, November 2007*.

[4] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Chapter 9)*. New York: Springer, 2006.

[5] J. S. Breese, D. Heckerman, and C. Kadie, *Empirical Analysis of Predictive Algorithms for Collaborative Filtering, In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98), pages 43–52, San Francisco, 1998. Morgan Kaufmann*. 1998.

[6] Salton , G. and Mc Gill, M.J., *Introduction to Modern Information Retrieval*. New York: Mc Graw-Hill Computer Series, 1983.

[7] P. Massa and P. Avesani, *Trust-aware Collaborative Filtering for Recommender Systems, Proceedings of the Federated International Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE, pages 492–508, 2004*.

[8] D. M. Pennock, E. Horvitz, S. Lawrence and C. L. Giles, *Collaborative Filtering by Personality Diagnosis: A Hybrid Memory and Model-Based approach. In UAI'00: Proc. 16th Conf. on Uncertainty in Artificial Intelligence, pages 473-480, Stanford, CA, 2000*.

[9] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, C. Kadie, *Dependency Networks for Density Estimation, Collaborative Filtering, and Data Visualization. Journal of Machine Learning Research. 1:49-75, 2000. Also appears as Technical Report MSR-TR-00-16, Microsoft Research, February, 2000*.

[10] Stijn Vanderlooy. Matlab Toolbox for Machine Learning, Technical Report MICC 08-03, Universiteit Maastricht, 2008. [Online]. http://www.personeel.unimaas.nl/s-vanderLooy//index.html

[11] X. Cui, T. E. Potok, and P. Palathingal, *Document Clustering using Particle Swarm Optimization, In Proceedings of the 2005 IEEE Swarm Intelligence Symposium, June, 2005, Pasadena, California, USA*. Pasadena, California, 2005.

[12] Eberhart, R. C., and Kennedy, J., *A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39-43. Piscataway, NJ: IEEE Service Center, 1995*.

[13] C. Abraham , P. A. Cornillon , E. Matzner-Lber and N. Molinari, *Unsupervised curve clustering using B-splines, Scandinavian J. Statist., vol. 30, no. 3, pp.581-595, 2003*.

[14] Storn, R., Price, K, *Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11: 341–359, 1997*.

[15] B. Schölkopf, C. Burges, and V. Vapnik, *Incorporating invariances in support vector learning machines, Artificial Neural Networks - ICANN'96, pages 47 - 52, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112*.

[16] I. Dhillon, Y. Guan and B. Kulis, *A fast kernel-based multilevel algorithm for graph clustering, Proc. 11th ACM Knowledge Discovery and Data Mining Conf., pp. 629-634, 2005*.

[17] De Bie, T., Cristianini, N., *Kernel methods for exploratory data analysis: a demonstration on text data, Proceedings of the joint IAPR international workshops on Syntactical and Structural Pattern Recognition, SSPR 2004 and Statistical Pattern Recognition, SPR 2004, Lisbon, 2004*.

[18] F.R. Bach and M.I. Jordan, *Learning spectral clustering. Neural Info. Processing Systems 16 (NIPS 2003), 2003*.

[19] McLachlan G, Peel, D. A, *Finite mixture models*. New York, 2000.